Die Kettenlinie
Theoretische Eigenschaften und Anwendung beim Bau eines frei stehenden Bogens

Facharbeit von Andreas Diewald

8. März 2024

Inhaltsverzeichnis

1 Einleitung

Ein Jupyter-Notebook, mit dem Python-Code zu dieser Arbeit, finden Sie hier:

`https://andreas-di.github.io/jupyter/lab/index.html?path=Kettenlinie.ipynb`

Die vorliegende Facharbeit und weitere Dateien, wie zum Beispiel die fertig druckbaren stl-Dateien, finden Sie hier:

`https://github.com/andreas-di/jupyter`.

Diese Facharbeit beschäftigt sich mit der mathematischen Kettenlinie. Die Kettenlinie (auch Katenoide genannt) ist die mathematische Kurve, die eine Kette beschreibt, die an zwei Punkten aufgehangen ist und nach Einwirkung der Schwerkraft in der Ruhelage hängt.

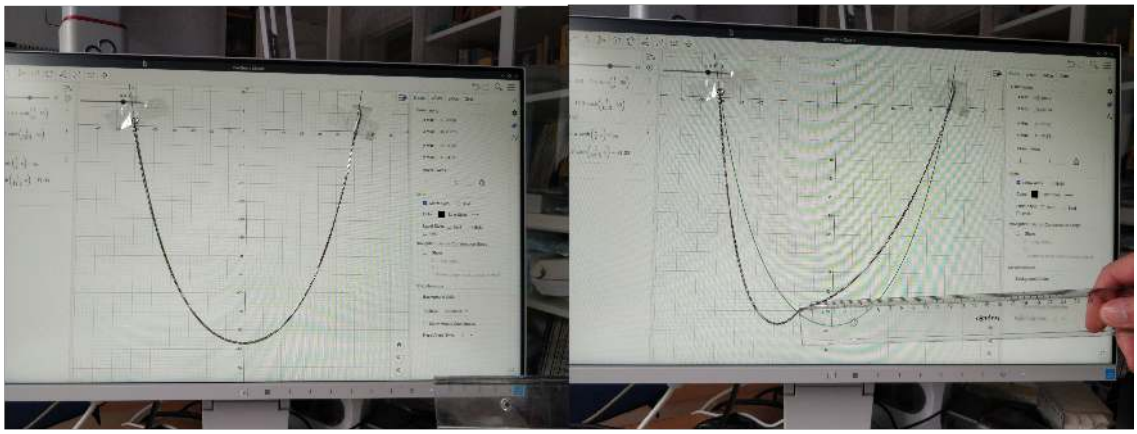


Abbildung 1: Eine Halskette ist mit Tesafilm am Bildschirm befestigt und man erkennt, dass ihre Form genau mit der mathematischen Katenoide übereinstimmt.

Ich werde auf die mathematische Beschreibung der Kettenlinie eingehen und diese anschließend an einem spezifischen Beispiel durch einen Bogen simulieren.
Dieser besteht aus einzelnen Klötzen ohne statische Verbindung, welche zu einem Bogen zusammengesetzt sind. Dieses Modell illustriert besonders deutlich die zentrale Eigenschaft der Kettenlinie, wonach in keinem Punkt seitliche Schubkräfte wirken.
Abschließend werde ich auf die historische Bedeutung und Anwendung im Bereich der Architektur eingehen.

Erst einmal folgt jedoch eine knappe Schilderung der Geschichte der Kettenlinie.

Geschichte

Galilei (1564 bis 1642) beschäftigte sich erstmals mit der Kettenlinie. Er vermutete, dass die Kettenlinie, zwar nicht gleich, aber ähnlich einer Parabel sei. Diese These widerlegte jedoch Joachim Jungius (1587 bis 1657) in seiner 1639 erschienenen „Geometria empyrica“.
Christian Huygens (1629 bis 1695) zeigt 1646 ebenfalls, dass es sich bei der Kettenlinie nicht um einem Parabel handeln könne. Da Huygens jedoch nicht in der Lage war, die Kettenlinie mathematisch zu beschreiben, bat er Leibniz darum.
Die exakte Herleitung der Kettenlinie erfolgte dann erstmals 1690 durch Gottfried Wilhelm Leibniz. Da Leibniz sich bei seiner Lösung unsicher war, rief er zum Wettbewerb auf, die Kettenlinie herzuleiten. Im Zuge dieses Wettbewerbs leiteten auch Huygens und Johann Bernoulli die mathematisch korrekte Beschreibung der Kettenlinie her.
Diese Lösungen wurden schließlich im Juni 1691 veröffentlicht.

Die Darstellung der geschichtlichen Erarbeitung der Kettenlinie ist einer Ausarbeitung von [Hajiabadi] entnommen.

Gliederung der Facharbeit

Der theoretischen Erarbeitung der Kettenlinie in Kapitel 5 und dem Bau des Bogens (Kapitel 6) sind die für den Bau benötigten Grundlagen vorangestellt (Kapitel 2 bis Kapitel 4).
Anschließend folgt die Herleitung der mathematischen Beschreibung der Kettenlinie in Kapitel 5 und die Dokumentation des Bogenbaus (Kapitel 6). Abschließend wird die historische Bedeutung der Kettenlinie in Kapitel 7 dargestellt.

2 Grundlagen: Herleitung der Bogenlänge

Die Bogenlänge ist die Länge, die man auf der Funktion zwischen zwei Punkten A und B zurücklegt (siehe Abbildung 2). Legt man beispielsweise eine Schnur über den Funktionsgraphen zwischen den beiden Punkten, ist die Länge der Schnur die Bogenlänge.

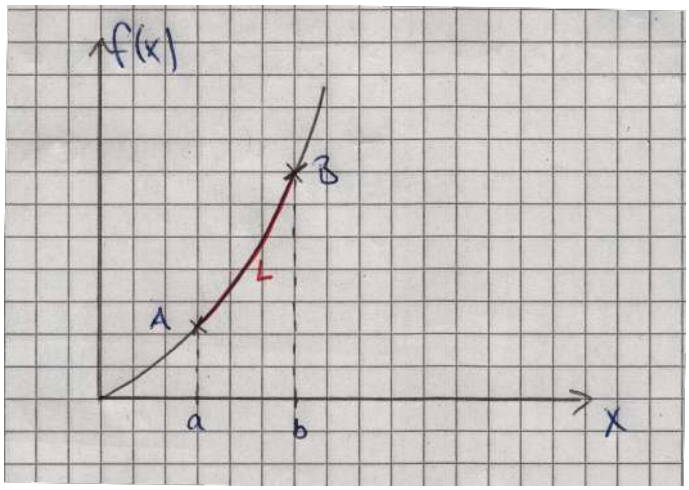


Abbildung 2: Bogenlänge L zwischen zwei Punkten

Um im Weiteren auf die Länge des freistehenden Bogens eingehen zu können, benötige ich die allgemeine Formel für die Bogenlänge zwischen zwei Punkten A und B.

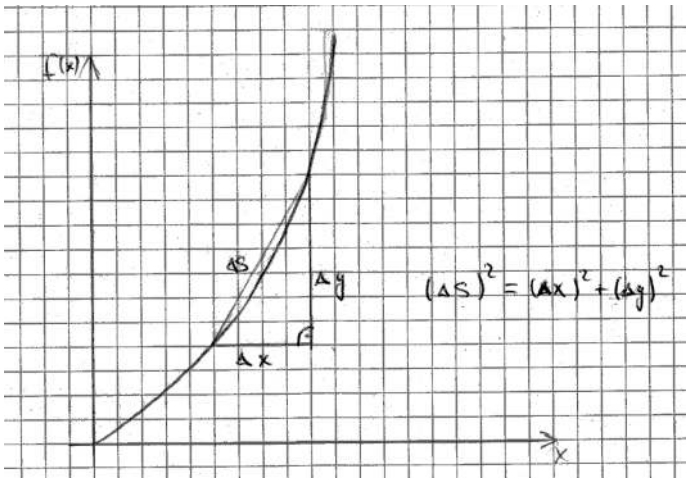


Abbildung 3: Ansatz Berechnung der Bogenlänge

Nach Abbildung 3 und dem Satz des Pythagoras gilt:

$$\begin{aligned}
(\Delta s)^2 &= (\Delta x)^2 + (\Delta y)^2 && | : (\Delta x)^2 \\
\left(\frac{\Delta s}{\Delta x}\right)^2 &= 1 + \left(\frac{\Delta y}{\Delta x}\right)^2 && |\sqrt{\cdot} \\
\frac{\Delta s}{\Delta x} &= \sqrt{1 + \left(\frac{\Delta y}{\Delta x}\right)^2} && | \cdot \Delta x
\end{aligned} \tag{1}$$

Für den Grenzfall $\Delta x \to 0$ gilt $\left(\frac{\Delta y}{\Delta x}\right)^2 = (f'(x))^2$, also

$$ds = \sqrt{1 + (f'(x))^2} \cdot dx \tag{2}$$

Die Bogenlänge L ist die Summe aller Längenelemente ds auf der Strecke zwischen A und B.

$$\begin{aligned}
L &= \sum \Delta s \\
&= \int_a^b ds \\
L &\overset{(2)}{=} \int_a^b \sqrt{1 + (f'(x))^2} dx
\end{aligned} \tag{3}$$

Die Formel für die Bogenlänge findet man unter anderem in [Formelsammlung, S. 24].

3 Drehmatrix

Da in einem späteren Abschnitt in Kapitel 6.7 Punkte in einem Koordinatensystem gedreht und verschoben werden, benötige ich noch Drematrizen. Daher werden diese zunächst noch behandelt.
Grundlagen im Bereich der Vektor-Matrix-Rechnung befinden sich im Anhang.

Der Vektor $\vec{v}$ soll gedreht um den Winkel α den Vektor $\vec{v\prime}$ ergeben (siehe Abbildung 4).

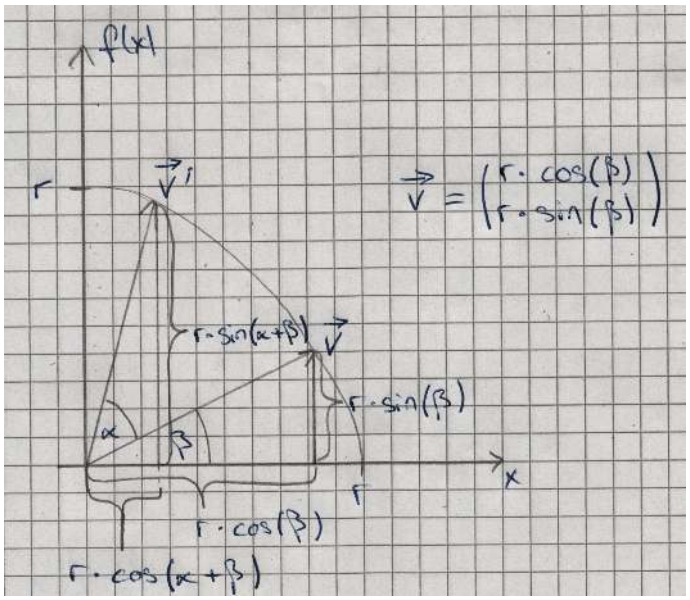


Abbildung 4: Der Vektor $\vec{v}$ wird mit der Drehmatrix gedreht. Der gedrehte Vektor ist hier bezeichnet mit $\vec{v'}$

$$\vec{v\prime} = \begin{pmatrix} r \cdot \cos(\alpha + \beta) \\ r \cdot \sin(\alpha + \beta) \end{pmatrix} \tag{4}$$

Mit den Additionstheoremen [Formelsammlung, S.14], [Müller-Fonfara, S. 318] für die trigonometrischen Funktionen

$$\begin{aligned} \sin(a + b) &= \sin(a) \cdot \cos(b) + \cos(a) \cdot \sin(b) \\ \cos(a + b) &= \cos(a) \cdot \cos(b) - \sin(a) \cdot \sin(b) \end{aligned} \tag{5}$$

ergibt sich aus Gleichung (4):

$$\vec{v\prime} \overset{(5)}{=} \begin{pmatrix} r \cdot \cos(\alpha) \cdot \cos(\beta) - \sin(\alpha) \cdot \sin(\beta) \\ r \cdot \sin(\alpha) \cdot \cos(\beta) + \cos(\alpha) \cdot \sin(\beta) \end{pmatrix} \tag{6}$$

Dies kann man als Matrix-Vektor-Multiplikation (siehe Anhang Gleichung (34)) schreiben:

$$\vec{v\prime} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \cdot \begin{pmatrix} r \cdot \cos(\beta) \\ r \cdot \sin(\beta) \end{pmatrix}$$

Wobei $\begin{pmatrix} r \cdot \cos(\beta) \\ r \cdot \sin(\beta) \end{pmatrix} = \vec{v}$ ist. Also gilt:

$$\vec{v\prime} = D \cdot \vec{v}$$

Mit D als Drehmatrix:

$$\boxed{D = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}} \tag{7}$$

Diese Formel für die Drehmatrix findet man in [Fischer, S. 203] oder auch [Athen/Bruhn, S. 208].

4 Grundlagen: Hyperbelfunktionen

Da die Kettenlinie durch eine cosh-Funktion beschrieben wird, benötige ich im Folgenden noch einige Grundlagen dazu. Daher werden dieses hier behandelt.

4.1 Cosinus hyperbolicus

Die Funktion cosh ist definiert als:

$$\cosh(x) := \frac{e^x + e^{-x}}{2} \qquad \text{[Formelsammlung, } S.25\text{]} \tag{8}$$

Graphische Veranschaulichung:

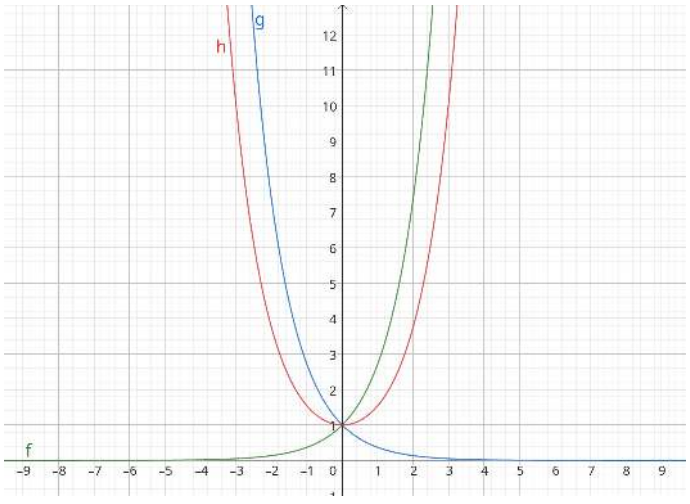


Abbildung 5: cosh als die „Hälfte “der Summe von e^x und e^{-x}

$$\begin{aligned} f(x) &= e^x \\ g(x) &= e^{-x} \end{aligned}$$

$$\begin{aligned} h(x) &= \frac{f(x) + (g(x))}{2} \\ &= \frac{e^x + e^{-x}}{2} \\ &= \cosh(x) \end{aligned}$$

Die cosh Funktion ist also die „Hälfte “der Summe von e^x und e^{-x}.

4.2 Sinus hyperbolicus

Die Funktion sinh ist definiert als:

$$\sinh(x) := \frac{e^x - e^{-x}}{2} \qquad \text{[Formelsammlung, } S.25\text{]} \tag{9}$$

Graphische Veranschaulichung:

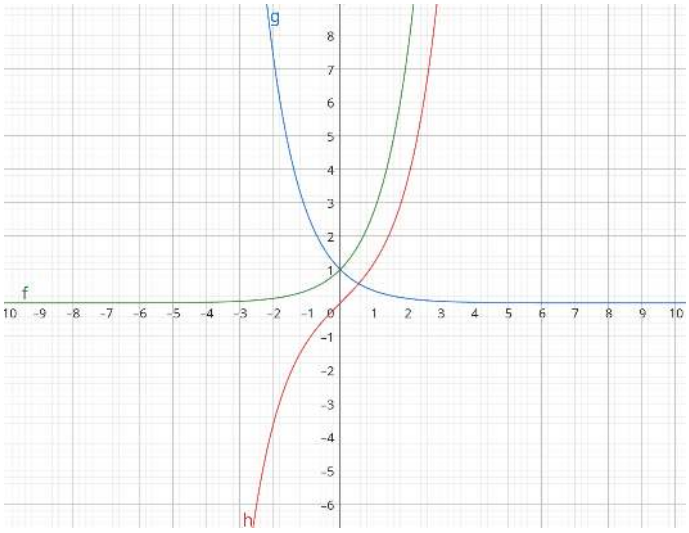


Abbildung 6: sinh als die „Hälfte “der Subtraktion von e^x und e^{-x}

$$\begin{aligned} f(x) &= e^x \\ g(x) &= e^{-x} \end{aligned}$$

$$\begin{aligned} h(x) &= \frac{f(x) - (g(x))}{2} \\ &= \frac{e^x - e^{-x}}{2} \\ &= \sinh(x) \end{aligned}$$

Die sinh Funktion ist also die „Hälfte “der Differenz von e^x und e^{-x}.

4.3 Ableitungen

Beim späteren Bau des freistehenden Bogen, werden die Ableitungen der cosh und sinh Funktion benötigt. Daher werden diese auch noch hergeleitet.

4.3.1 Cosinus hyperbolicus

Mit der Kettenregel erhält man:

$$\begin{aligned}\cosh'(x) &= \left(\frac{e^x + e^{-x}}{2}\right)' \\ &= \frac{(e^x)' + (e^{-x})'}{2} \\ &= \frac{e^x + e^{-x} \cdot (-1)}{2} \\ &= \frac{e^x - e^{-x}}{2}\end{aligned}$$

Da $\frac{e^x - e^{-x}}{2} = \sinh(x)$ ist, gilt:

$$\cosh'(x) = \sinh(x) \tag{10}$$

4.3.2 Sinus hyperbolicus

Mit der Kettenregel eribt sich:

$$\begin{aligned}\sinh'(x) &= \left(\frac{e^x - e^{-x}}{2}\right)' \\ &= \frac{(e^x)' - (e^{-x})'}{2} \\ &= \frac{e^x - e^{-x} \cdot (-1)}{2} \\ &= \frac{e^x + e^{-x}}{2}\end{aligned}$$

Da $\frac{e^x + e^{-x}}{2} = \cosh(x)$ ist, gilt:

$$\sinh'(x) = \cosh(x) \tag{11}$$

4.4 Zusammenhänge zwischen cosh und sinh

Behauptung:

$$\cosh^2(x) - \sinh^2(x) = 1 \qquad \text{[Formelsammlung, } S.25\text{]} \tag{12}$$

Beweis:

$$\begin{aligned} \cosh^2(x) - \sinh^2(x) &= \left(\frac{e^x + e^{-x}}{2}\right)^2 - \left(\frac{e^x - e^{-x}}{2}\right)^2 \\ &= \frac{e^{2x} + 2 \cdot (e^x \cdot e^{-x}) + e^{-2x}}{4} - \frac{e^{2x} - 2 \cdot (e^x \cdot e^{-x}) + e^{-2x}}{4} \end{aligned}$$

Da $(e^x \cdot e^{-x}) = 1$ ist:

$$\begin{aligned} &= \frac{e^{2x} + 2 + e^{-2x}}{4} - \frac{e^{2x} - 2 + e^{-2x}}{4} \\ &= \frac{e^{2x} + 2 + e^{-2x} - e^{2x} + 2 - e^{-2x}}{4} \\ &= \frac{2 + 2}{4} \\ &= 1 \qquad\qquad q.e.d \end{aligned}$$

5 Grundlagen Kettenlinie

Die Kettenlinie beschreibt die Linie, in der eine Kette, aufgehangen in zwei Punkten, in der Ruhelage hängt (siehe Abbildung 1 aus der Einleitung).

Eine hängende Kette hat die Eigenschaft, dass in jedem Punkt der Kette nur tangentiale Kräfte wirken. Die seitliche Schubkräfte stehen im Gleichgewicht.
Im folgenden Unterkapitel 5.1 wird hergeleitet, dass sich eine Kettenlinie durch die Funktion

$$f(x) = a \cosh\left(\frac{1}{a}x\right) + y_0$$

darstellen lässt.

5.1 Herleitung Kettenlinie

Die Herleitung folgt der Darstellung von [Stifel]. Außerdem ist die Herleitung bei [Lern-Helfer] zu finden. Einer ausführlichere Ausarbeitung ist auch in [Glück, S. 2] dargestellt.

Betrachtet wird hier eine Kette, welche an zwei Punkten aufgehangen ist und in der Ruhelage hängt.
In Abbildungen 7 wird die Kettenlinie durch einzelne Massepunkte, die miteinander in Verbindung stehen, veranschaulicht. Die Kettenlinie wird hier „vergrößert“dargestellt, indem nur einzelne Massepunkte eingezeichnet sind.

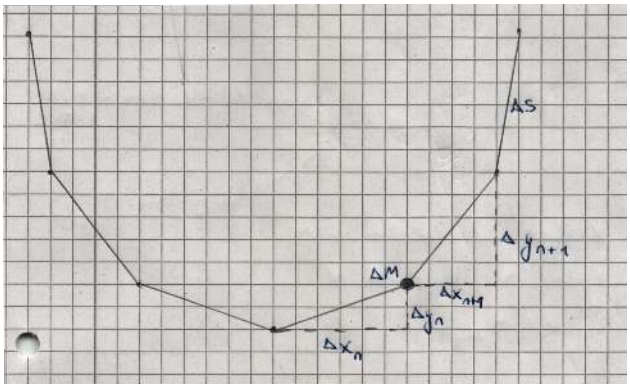


Abbildung 7: Kettenlinie simuliert durch einzelne Massepunkte

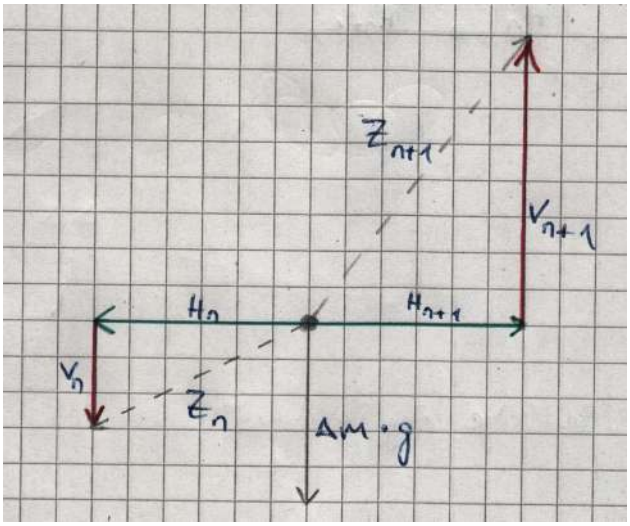


Abbildung 8: Einzelner Massepunkt auf der Kettenlinie

Betrachtet man einen einzelnen Massepunkt, wirken auf diesen die Kräfte Z, sowie die Gewichtskraft $m \cdot g$.
Die Z Kräfte lassen sich jeweils in vertikale Kraftkomponenten V und horizontale Kraftkomponenten H unterteilen (siehe Abbildung 8).
Daher gilt bei einer Kette im Gleichgewicht:

$$H_n = H_{n+1} = konstant = H \tag{13}$$
$$V_{n+1} = V_n + \Delta m \cdot g \tag{14}$$

Gleichung (13) bedeutet, dass keine resultierenden seitlichen Schubkräfte wirken. Denn wenn welche wirken würden, würde sich die Form der Kette solange nach diesen ändern, bis keine mehr wirken würden.
Nur wenn Gleichung (13) gilt, hängt die Kette in Ruhelage.

Für die Steigung der linken Seite des betrachteten Massepunkts gilt:

$$\frac{\Delta y_n}{\Delta x_n} = \frac{V_n}{H_n} \tag{15}$$

Und auf rechter Seite:

$$\frac{\Delta y_{n+1}}{\Delta x_{n+1}} = \frac{V_{n+1}}{H_{n+1}} \tag{16}$$

Da in der Ruhelage die vertikale Kraftkomponente nach unten aus der Gewichtskraft und der Teilkraft der entlang der Kette wirkenden Zugkraft entsteht (siehe Formel (14)) kann man Gleichung (16), auch schreiben als:

$$\begin{aligned} \frac{\Delta y_{n+1}}{\Delta x_{n+1}} &= \frac{V_n + \Delta m \cdot g}{H} \\ &= \frac{V_n}{H} + \frac{\Delta m \cdot g}{H} \\ &\overset{(15)}{=} \frac{\Delta y_n}{\Delta x_n} + \frac{\Delta m \cdot g}{H} \end{aligned}$$

Also ist die Steigung auf der rechten Seite gleich der Summe aus der Steigung auf der linken Seite und $\frac{\Delta m \cdot g}{H}$.

$$\frac{\Delta y_{n+1}}{\Delta x_{n+1}} - \frac{\Delta y_n}{\Delta x_n} := \Delta\left(\frac{\Delta y}{\Delta x}\right)$$

Also ist die Steigung an einem beliebigen Punkt:

$$\Delta\left(\frac{\Delta y}{\Delta x}\right) = \frac{\Delta m \cdot g}{H} \tag{17}$$

Allgemein gilt:

$$\Delta m = \rho \cdot \Delta s \tag{18}$$

Wobei ρ die Masse pro Länge ist.
Nach Einsetzten in Gleichung (17):

$$\Delta\left(\frac{\Delta y}{\Delta x}\right) \overset{(18)}{=} \frac{\rho \cdot \Delta s \cdot g}{H}$$

Durch Einsetzten von $\Delta s = \sqrt{1 + (f'(x))^2} \cdot \Delta x$ (siehe Grundlagen: Formel (3)), erhält man:

$$\begin{aligned} \Delta\left(\frac{\Delta y}{\Delta x}\right) &\overset{(3)}{=} \sqrt{1 + \left(\frac{\Delta y}{\Delta x}\right)^2} \cdot \frac{\rho \cdot g}{H} \cdot \Delta x \qquad | : \Delta x \\ \frac{\Delta\left(\frac{\Delta y}{\Delta x}\right)}{\Delta x} &= \sqrt{1 + \left(\frac{\Delta y}{\Delta x}\right)^2} \cdot \frac{\rho \cdot g}{H} \end{aligned} \tag{19}$$

y ist Funktion von x:

$$\frac{\Delta y}{\Delta x} = \frac{\Delta y(x)}{\Delta x} \underset{\lim \Delta x \to 0}{\longrightarrow} \frac{dy(x)}{dx} = y'(x)$$

Also:

$$\frac{\Delta\left(\frac{\Delta y}{\Delta x}\right)}{\Delta x} \underset{\lim \Delta x \to 0}{\longrightarrow} \frac{\Delta y'}{\Delta x} \underset{\lim \Delta x \to 0}{\longrightarrow} y''(x)$$

Also wird aus (19) die Differentialgleichung:

$$\boxed{y''(x) = \sqrt{1 + (y'(x))^2} \cdot \frac{\rho \cdot g}{H}} \tag{20}$$

Ansatz zum Lösen der Differenzialgleichung:

$$y(x) = a \cdot \cosh\left(\frac{1}{a}x\right) + y_0 \tag{21}$$

Einsetzten von Ansatz (21) in die Differenzialgleichung (20):

$$\left(a \cdot \cosh\left(\frac{1}{a}x\right) - y_0\right)'' \overset{(21)}{=} \sqrt{1 + \left(\left(a \cdot \cosh\left(\frac{1}{a}x\right) - y_0\right)'\right)^2} \cdot \frac{\rho \cdot g}{H}$$

Nach Ableiten der Funktion (siehte Formel (10) und Formel (11)):

$$\begin{aligned}\left(\sinh\left(\frac{1}{a}x\right)\right)' &= \sqrt{1+\sinh^2\left(\frac{1}{a}x\right)}\cdot\frac{\rho\cdot g}{H}\\ \frac{1}{a}\cdot\cosh\left(\frac{1}{a}x\right) &= \sqrt{1+\sinh^2\left(\frac{1}{a}x\right)}\cdot\frac{\rho\cdot g}{H}\end{aligned}$$

Durch Beziehung $\cosh^2-\sinh^2=1$ (siehe Grundlagen Hyperbelfunktionen, Gleichung (12) und Wurzel ziehen:

$$\frac{1}{a}\cdot\cosh\left(\frac{1}{a}x\right) = \cosh\left(\frac{1}{a}x\right)\cdot\frac{\rho\cdot g}{H}$$

Wenn $\frac{1}{a}=\frac{\rho\cdot g}{H}$ wird die Differentialgleichung also erfüllt durch:

$$y(x)=a\cdot\cosh\left(\frac{1}{a}x\right)+y_0$$

Im Folgenden werde ich die Bezeichung $f(x)$ für $y(x)$ benutzen.

$$\boxed{f(x)=a\cdot\cosh\left(\frac{1}{a}x\right)+y_0} \tag{22}$$

6 Freistehender Bogen

Hängt man an eine hängende Kette eine bestimmte Masse, bildet sich so die Kette so aus, dass keine seitlichen Schubkräfte wirken. Dreht man dieses Modell um, erhält man den optimalen Bogen, um eine solche Masse zu tragen. Das liegt daran, dass genau wie bei der Kette in keinem Punkt seitliche Schubkräfte wirken. Diese Eigenschaft, ermöglichte das verwenden von Hängemodellen in der Architektur (siehe Kapitel 7).

Dies gilt auch dann, wenn keine Masse auf dem Bogen lasten soll. Dann ist der optimale stabile Bogen die umgedrehte Kettenlinie.

Um zu veranschaulichen, dass bei einem solchen Bogen keine seitlichen Schubkräfte wirken, baue ich im Folgenden einen Bogen aus einzelnen Klötzen, die nicht statisch miteinander verbunden sind (siehe Abbildung 9). Würden seitliche Schubkräfte wirken, würden sich die Klötze zur Seite weg schieben und der Bogen würde zusammenbrechen.

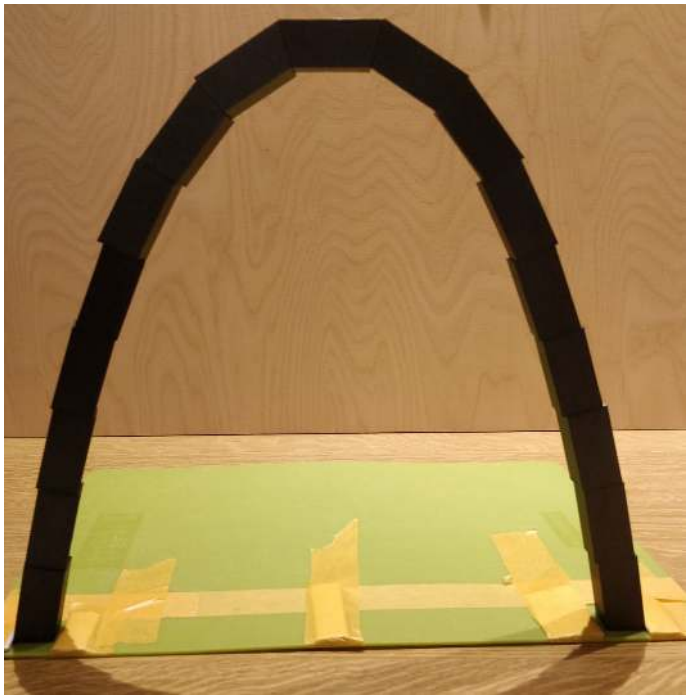


Abbildung 9: Freistehender Bogen aus einzelnen 3D-gedruckten Klötzen. Die Klötze stehen lose aufeinander und sind nicht miteinander verbunden.

Der Bogen wird auf dem Boden von zwei festgeklebten Erhöhungen festgehalten (siehe Abbildung 9). Das ist nötig, da an diesen stellen die Kraft von den oberen Klötzen nicht komplett abgefangen wird (siehe Abbildung 10).

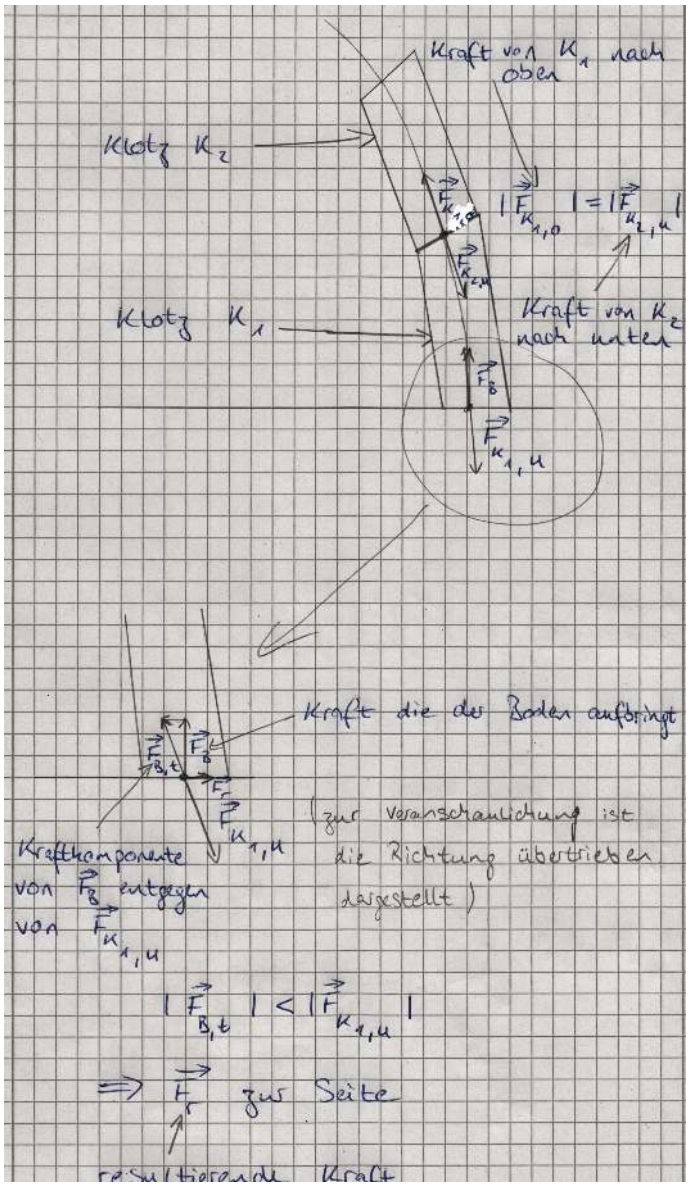


Abbildung 10: Die tangentiale Kraftkomponente $\vec{F}_{B,t}$ der vom Boden aufgebracht Kraft $\vec{F}_B$ ist kleiner als die tangentiale $\vec{F}_{K_{1,u}}$. Daher ist die resutierende Kraft zur Seite gerichtet.

Da auf dem Bogen keine Masse lasten soll, wird dieser durch eine Kettenlinie beschrieben. Vor dem Bau des Bogens wird die Länge L und Breite B am Boden des Bogens festgelegt (siehe Abbildung 11).

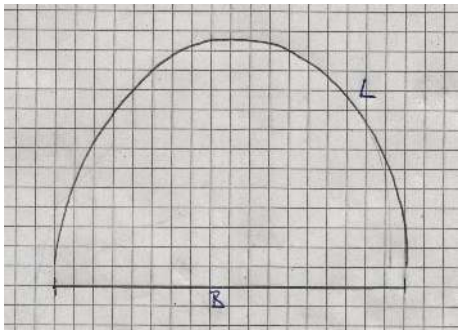


Abbildung 11: L ist die Bogenlänge

Auch die Breite b und Höhe h der Klötze (siehe Abbildung 12) werden festgelegt.

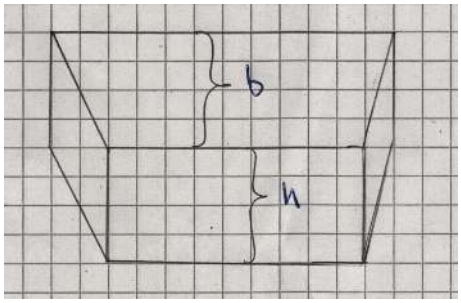


Abbildung 12: b ist die Breite des Klotzes, h seine Höhe

Hier wurden die Werte gewählt als:

$$\begin{aligned} L &= 136LE \\ B &= 60LE \\ b &= 4LE \\ h &= 4LE \end{aligned}$$

Die bogenbeschreibende Funktionsgleichung soll so gewählt werden, dass der Teil unterhalb der x-Achse den Bogen beschreibt, (siehe Abbildung 13).
Daher gilt, dass die Nullstelle der Funktion $x_0 = \frac{B}{2} = 30LE$ und die Bogenlänge zwischen den Nullstellen $-x_0$ und x_0 gleich L ist.

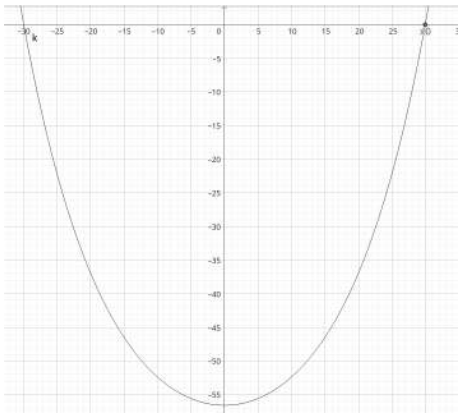


Abbildung 13: Bogenbeschreibende Kettenlinie

Ich habe mich dafür entschieden, den Bogen aus 17 Klötzen zu bauen.
Durch die ungerade Anzahl, ist der Bogen stabiler, da der obere Klotz waagerecht liegt (siehe Abbildung 14).

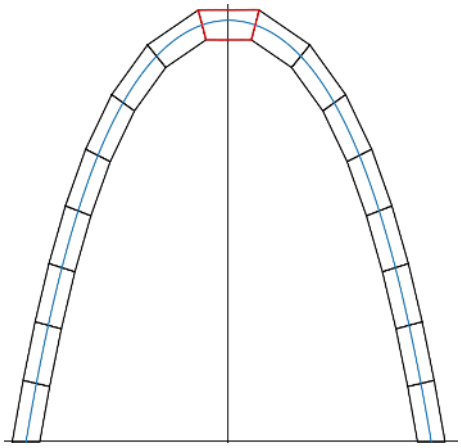


Abbildung 14: Oberster Klotz liegt waagerecht

6.1 Bogenlänge Kettenlinie

Um die Maße der einzelnen Klötze bestimmen zu können, muss im ersten Schritt die Funktionsgleichung der Kettenlinie bestimmt werden, die den Bogen simuliert.

Die bogenbeschreibende Funktion ist so ins Koordinatensystem gelegt, dass sie Achsensymmetrie aufweist und der Teil unterhalb der x-Achse den Bogen beschreibt (siehe Abbildung 15).

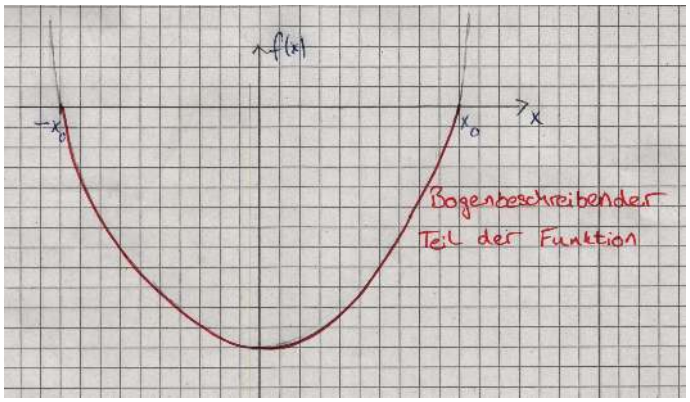


Abbildung 15: Teil der Kettenlinie, der den Bogen simuliert

Um den Faktor a der Kettenlinie $f(x) = a \cdot \cosh\left(\frac{1}{a}x\right) - y_0$ zu bestimmen, wird zuerst die Gleichung für die Bogenlänge L unterhalb der x-Achse aufgestellt.

Aus Abbildung 15 folgt, dass die Bogenlänge zwischen den beiden Nullstellen $-x_0$ und x_0 gleich der Länge L des Bogens sein muss. Mit Formel (3) für die allgemeine Bogenlänge gilt:

$$L = \int_{-x_0}^{x_0} \sqrt{1 + (f'(x))^2}\, dx$$

Da die Kettenlinie durch eine cosh Funktion beschrieben wird ($f(x) = a \cdot \cosh\left(\frac{1}{a}x\right)$ siehe Formel (22)), und diese abgeleitet $f'(x) = \sinh\left(\frac{1}{a}x\right)$ ergibt (siehe Formel (10)), wird aus dieser Gleichung:

$$L \overset{(10)}{=} \int_{-x_0}^{x_0} \sqrt{1 + \sinh^2\left(\frac{1}{a}x\right)}\, dx \tag{23}$$

Durch die Beziehung $\cosh^2(x) - \sinh^2(x) = 1$ (siehe Grundlagen Gleichung (12)), kann man Gleichung(23) auch schreiben als:

$$\begin{aligned} L &\overset{(12)}{=} \int_{-x_0}^{x_0} \sqrt{\cosh^2\left(\frac{1}{a}x\right)}\, dx \\ &= \int_{-x_0}^{x_0} \cosh\left(\frac{1}{a}x\right) dx \\ &\overset{(11)}{=} \left[a \cdot \sinh\left(\frac{1}{a}x\right)\right]_{-x_0}^{x_0} \\ &= a \cdot \sinh\left(\frac{1}{a}x_0\right) - \left(a \cdot \sinh\left(-\frac{1}{a}x_0\right)\right) \end{aligned} \tag{24}$$

Da die sinh Funktion punktsymmetrisch ist, gilt $a \cdot \sinh\left(-\frac{1}{a}x_0\right) = -\left(a \cdot \sinh\left(\frac{1}{a}x_0\right)\right)$. Und da $(-1) \cdot (-1) = +1$ ist, kann man Gleichung (24) auch schreiben als:

$$L = 2a \cdot \sinh\left(\frac{1}{a}x_0\right) \tag{25}$$

Alle Werte bis auf a sind konstant. ($x_0 = 30LE$ und $L = 136LE$)
Mit Hilfe dieser Gleichung wird im nachfolgenden Kapitel der Fakor a der Funktion bestimmt.

6.2 Bestimmung Faktor a

Zur Beschreibung einer konkreten Kettenlinie muss der Faktors a der Funktionsgleichung (Gleichung (22)) für die Kettenlinie bestimmt werden.
$L = 136LE = konstant$
$x_0 = 30LE = konstant$

$$\begin{aligned} L &\overset{(25)}{=} 2a \cdot \sinh\left(\frac{1}{a}x_0\right) && | - 2a \cdot \sinh\left(\frac{1}{a}x_0\right) \\ 0 &= L - 2a \cdot \sinh\left(\frac{1}{a}x_0\right) \end{aligned} \tag{26}$$

Betrachtet man nun $g(a) := L - 2a \cdot \sinh\left(\frac{1}{a}x_0\right)$ als Funktion von a, sind die Nullstellen dieser Funktion gesucht.
Der Wert a wird, mit einem Python-Programm (siehe Anhang) mittels Intervallhalbierung, numerisch bestimmt, da es nicht möglich ist, Gleichung (26) algebraisch nach a umzustellen.

Mit den von mir verwendeten Parametern ($L = 136LE$, $x_0 = 30LE$) ergibt sich mit Hilfe des Python-Programmes die Nullstelle:

$$\boxed{a \approx 12,54}$$

Der Wert a ist auf zwei Nachkommastellen gerundet. (Genauere Angaben befinden sich im Anhang.)

Damit sind nun alle Konstanten in der Kettenlinie (22) bestimmt. Der Faktor y_0 wurde ermittelt, indem die Funktion der Kettenlinie an der Nullstelle x_0 gleich Null gesetzt wurde. Dieser Ausdruck wurde nach y_0 umgestellt ($y_0 = -a \cdot \cosh\left(\frac{1}{a}x_0\right)$). Da a und x_0 bekannt sind, kann y_0 bestimmt werden.

$$\boxed{y_0 \approx -69.15}$$

Auch dieser Wert wurde mittels des Pythonprogramms bestimmt und ist auf zwei Nachkommastellen gerundet.

6.3 Bestimmung x Wert bei gegebener Bogenlänge auf der Funktion

Hier wird die Kettenlinie erstmalig in gleich lange Stücke unterteilt. Dieses Stücke werden später durch Klötze simuliert.
Die gesamte Bogenlänge wird durch Formel (23): $L = \int_{-x_0}^{x_0} \sqrt{1 + \sinh^2\left(\frac{1}{a}x\right)}dx$ bestimmt.

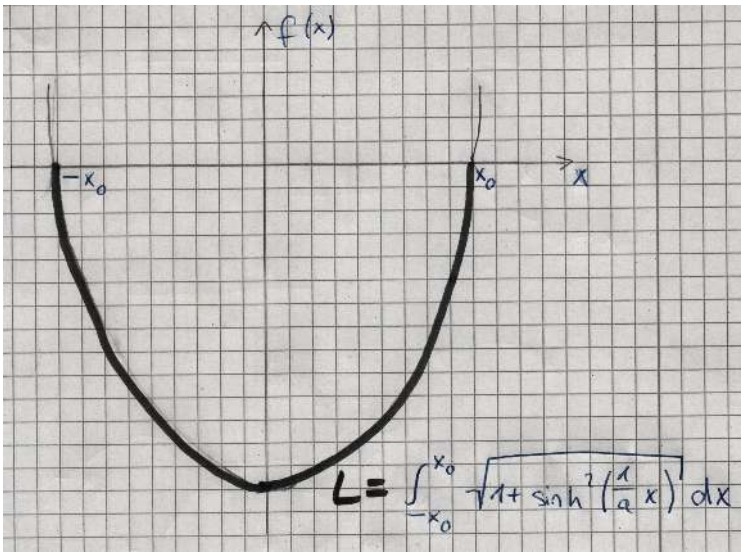


Abbildung 16: Bogenlänge zwischen den Nullstellen

Also wird die Bogenlänge von 0 bis zu einer Stelle x durch das Integral von 0 bis x bestimmt.

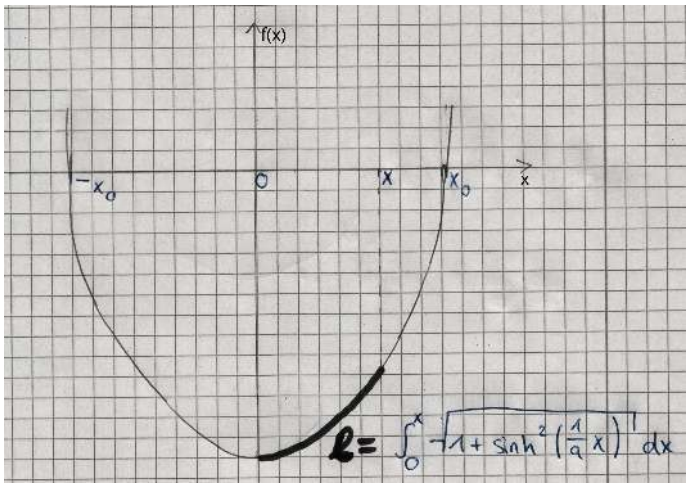


Abbildung 17: Bogenlänge zwischen der Stelle 0 und x

$$l \overset{!}{=} \int_0^x \sqrt{1 + (f'(x))^2} dx = \int_0^x \sqrt{1 + \sinh^2\left(\frac{1}{a}x\right)} dx$$

Mit der Beziehung $\cosh^2 - \sinh^2 = 1$ (siehe Formel (12)):

$$\begin{aligned} l &\overset{(12)}{=} \int_0^x \cosh\left(\frac{1}{a}x\right) dx \overset{(11)}{=} \left[a \cdot \sinh\left(\frac{1}{a}x\right)\right]_0^x \\ &= a \cdot \sinh\left(\frac{1}{a}x\right) - a \cdot \sinh\left(\frac{1}{a} \cdot 0\right) \end{aligned}$$

$$\left[a \cdot \sinh\left(\frac{1}{a} \cdot 0\right) = 0\right] \tag{27}$$

$$\begin{aligned} &\overset{(27)}{=} a \cdot \sinh\left(\frac{1}{a}x\right) && | : a \\ \frac{l}{a} &= \sinh\left(\frac{1}{a}x\right) && | \quad arcsinh(\cdot) \\ arcsinh\left(\frac{l}{a}\right) &= \frac{1}{a} \cdot x && | \cdot a \\ x &= a \cdot arcsinh\left(\frac{l}{a}\right) \end{aligned} \tag{28}$$

Mit dieser Formel lässt sich ein x-Wert x in Abhängigkeit von eine Bogenlänge l bestimmen. Also ist x eine Funktion von l, $x = x(l) = a \cdot arcsinh\left(\frac{l}{a}\right)$.
Die Konstante a ist hier $a \approx 12,5$ (wie in Kapitel 6.2 bestimmt).

Die Bogenlänge L der Funktion wird in gleichlange Teile unterteilt (siehe Abbildung 18), wobei jeder dieser Teile durch einen Klotz des Bogens simuliert werden soll.

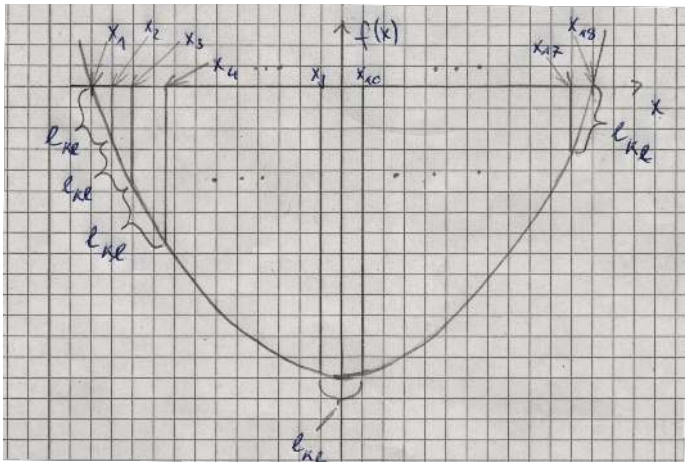


Abbildung 18: Bogenlänge unterteilt in Teillänge l_{Kl}

Die durch einen Klotz beschriebene Bogenlänge l_{Kl}, ist also die gesamte Länge L des Bogens geteilt durch die Anzahl der Klötze.
Für die Anzahl der Klötze habe ich eine ungerade Anzahl $anzahl_k = 17$ gewählt, sodass der oberste Klotz des Bogens waagerecht liegt (siehe Abbildung 14).
Da ich $L = 136LE$ gewählt habe, ist die durch einen Klotz beschriebene Bogenlänge hier $l_{Kl} = \frac{136LE}{17} = 8LE$.
Um die Liste der x-Werten zu erhalten muss l_1 nun bei $-\frac{L}{2}$ starten und in jedem Schritt um l_{Kl} erhöht werden, bis $l_{18} = \frac{L}{2}$ (siehe Abbildung 19).
Diese Werte für l_n werden in die Formel (28) für einen x-Wert in Ahängigkeit der Bo-

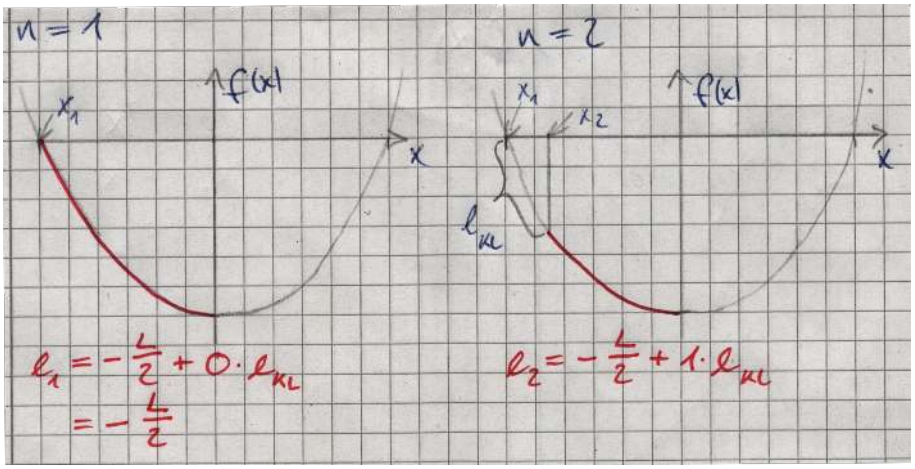


Abbildung 19: Bogenlänge l_n für $n = 1$ und $n = 2$

genlänge l eingesetzt.
Also gilt für den n-ten x-Wert

$$x_n = a \cdot arcsinh\left(\frac{l_n}{a}\right)$$

mit $a \approx 12,5$ und

$$l_n = -\frac{L}{2} + (n-1) \cdot l_k \tag{29}$$

wobei

$$n = 1, 2, 3..., 18$$

Also ist für $n = 1$ die zugehörige Bogenlänge $l_1 = -\frac{L}{2}$ und $x_1 = -x_0$. Das gleiche gilt für $n = 18$ auf positiver Seite. Also $l_{18} = \frac{L}{2}$ und somit $x_{18} = x_0$.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
l_n	-68.0	-60.0	-52.0	-44.0	-36.0	-28.0	-20.0	-12.0	-4.0	4.0	12.0	20.0	28.0	36.0	44.0	52.0	60.0	68.0
x_i	-30.0	-28.5	-26.7	-24.7	-22.3	-19.4	-15.6	-10.7	-3.9	3.9	10.7	15.6	19.4	22.3	24.7	26.7	28.5	30.0
$f(x_i)$	0.0	-7.9	-15.7	-23.4	-31.0	-38.5	-45.5	-51.8	-56.0	-56.0	-51.8	-45.5	-38.5	-31.0	-23.4	-15.7	-7.9	0.0

Die Werte in der Tabelle habe ich mittels einer for-Schleife in einem Pythonprogramm ermittelt (siehe Anhang).
Die x- und $f(x)-$Werte werden benötigt, um die Längen der Klötze und die Schneidewinkel an den Seiten der Klötze zu bestimmen.

6.4 Bestimmung der Schneidewinkel

Um die Klötze zu bauen werden die Schneidewinkel benötigt, unter denen sich die Klötze berühren. Dieses werden in diesem Kapitel berechnet.
Die Klötze liegen so auf der Funktion, dass der Funktionsgraph genau durch die Mitte der Ränder des Klotzes geht (siehe Abbildung 20). An diesen Stellen befinden sich die Punkte der x-Werte (siehe Abbildung 20).

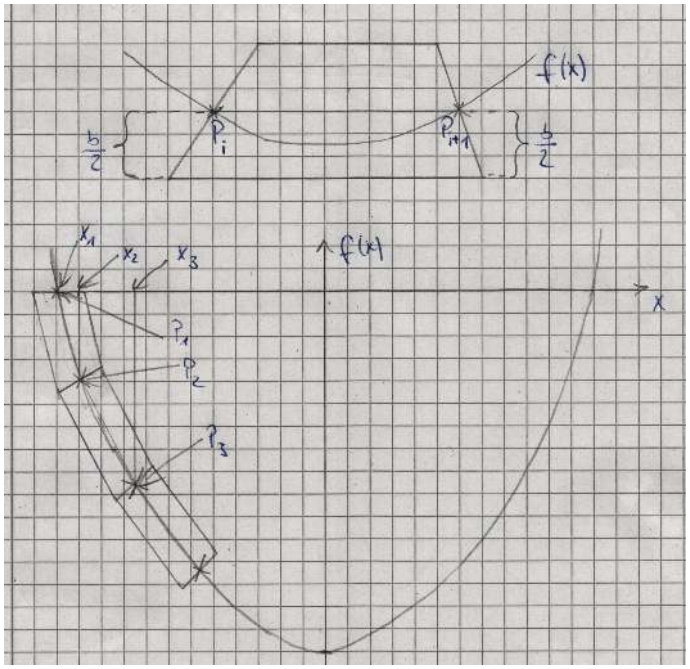


Abbildung 20: Lage des Funktionsgraphen in einem Klotz

Um den optimalen Bogen zu erhalten, müssten die Schneidewinkel in einem 90°-Winkel auf den wirkenden Kräften stehen. Da hier die Kräfte tangential wirken (siehe Grundlagen Kettenlinie), wären die korrekten Schneidewinkel die Normalen-Steigungswinkel.

Hätten die Klötze keine Breite, wäre es möglich diese als Schneidewinkel zu verwenden. Doch da die Klötze eine Breite haben, würden die Klötze nicht genau aufeinander aufliegen, würde man die Normalen-Steigungswinkel verwenden, (siehe Abbildung 21).

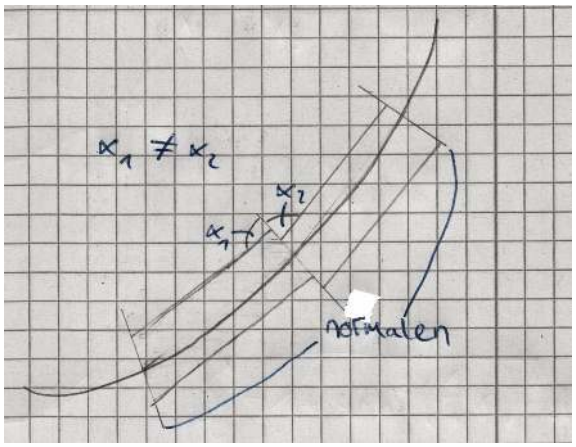


Abbildung 21: Lage der Klötze mit Normalen als Schnitte. Die Klötze würden nicht genau aufeinander aufliegen.

Da die Außenseiten der Klötze parallel zu der Sekanten durch die beiden x-Werte sind (siehe Abbildung 20), liegen die Klötze genau aufeinander auf, wenn der Winkel zwischen diesen gleich dem zwischen den jeweiligen Sekanten ist.

Der Schneidewinkel λ ist dementsprechend die Hälfte des Winkels γ zwischen den jeweiligen Sekanten, (siehe Abbildung 22).

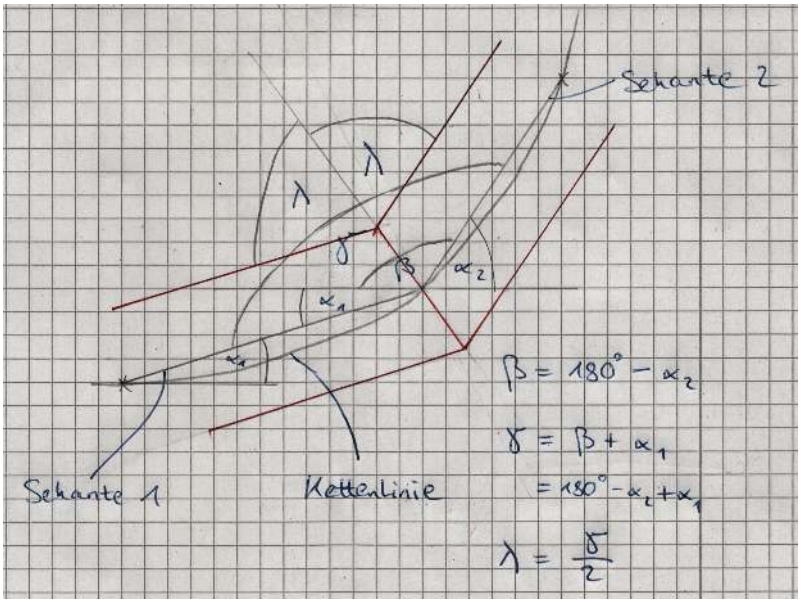


Abbildung 22: Winkel $\lambda = \frac{180° - \alpha_2 + \alpha_1}{2}$ zwischen den Klötzen (siehe Formel (30))

Also gilt für den Schneidewinkel λ zwischen zwei Klötzen:

$$\lambda = \frac{180° - \alpha_2 + \alpha_1}{2} \tag{30}$$

Wobei α_1 und α_2 die Sekanten-Steigungswinkel der jeweiligen Sekanten aus Abbildung 22 sind.

Besonderheit bei dem ersten und letzten Klotz

Da die Schneidewinkel λ jeweils von zwei Sekantensteigungen α abhängig sind, würde man für den ersten und letzten Klotz keinen Schneidewinkel an der x-Achse erhalten, da dort ein weiterer Sekanten-Steigungswinkel fehlt.
Da mein Python-Programm aus den Sekantensteigungswinkeln automatisch die Schneidewinkel und daraus die Klotzmaße ermittelt, wäre es umständlich, in jedem Schritt den ersten und letzten Klotz gesondert zu behandeln.
Um dies zu umgehen, habe ich den ersten und letzten Klotz an der x-Achse gespiegelt (siehe Abbildung 23) und die Sekantensteigungswinkel, die sich für diese gespiegelten Sekanten ergeben an die erste und letzte Stelle der Liste gestellt. Für den ersten und letzten Schneidewinkel λ_1 und λ_{18} gilt also $\lambda_1 = \alpha_1$ und $\lambda_{18} = \alpha_{17}$ (siehe Abbildung 23)

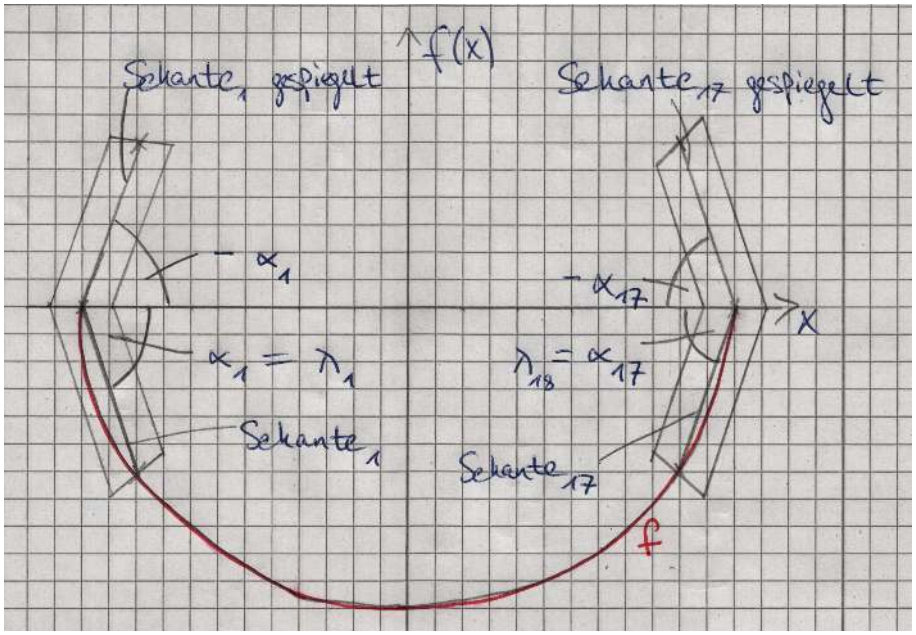


Abbildung 23: Erster und letzter Klotz an der x-Achse gespiegelt

Mit diesem Trick, können alle anderen Funktionen meines Python-Programmes wie gewohnt über die als Parameter benötigten Listen iterieren, ohne Ausnahmen beim ersten und letzten Klotz machen zu müssen. Andernfalls müsste in der Liste der Schneidewinkel an ersten und letzter Stelle der jeweilige Sekanten-Steigungswinkel angegeben werden.

Numerische Ergebnisse

In der nachfolgenden Tabelle sind die verwendeten Schneidewinkel λ dargestellt. Die Werte in der Tabelle wurden genau wie die x-Werte mit dem Pythonprogramm berechnet (siehe Anhang).

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
λ	78.9	89.23	88.98	88.61	87.98	86.86	84.66	80.19	74.04	74.04	80.19	84.66	86.86	87.98	88.61	88.98	89.23	78.9

6.5 Bestimmung der Klotz-Ausmaße

Zum Drucken der Klötze werden die genauen Abmaße der Klötze benötigt. Diese werden in diesem Unterkapitel berechnet.

6.5.1 Länge der Sekante

Die Sekante, die den Klotz darstellt, verläuft an dessen Seiten genau durch die Mitten (siehe Abbildung 20). Also genau durch die Punkte, die den Klotz in seiner Lage festlegen. Jede Sekante ist, wie der zugehörige Klotz von zwei Punkten und somit den zwei x-Werten abhängig (siehe Abbildung 24). Der Index der Sekanten gibt an, durch welche Punkte diese verläuft (beispielsweise verläuft die Sekante $s_{1,2}$ durch den Punkt mit dem zugehörigen x-Wert x_1 und den Punkt mit x-Wert x_2):

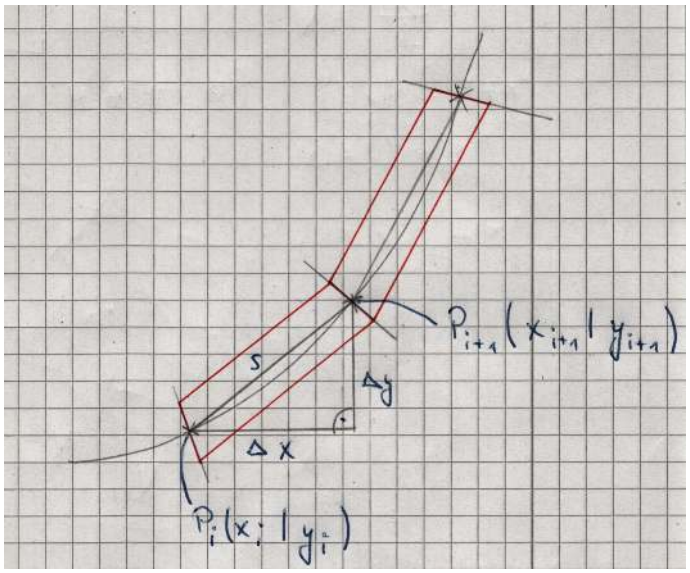


Abbildung 24: Länge der Sekante

$$\begin{aligned} j &= i+1 \\ \Delta x &= x_j - x_i \\ \Delta y &= y_j - y_i \end{aligned} \tag{31}$$

Da die Koordinaten aller Punkte auf der Funktion bekannt sind (siehe Tabelle aus Kapitel 6.3), kann die Sekanten-Länge s nach dem Satz von Pythagoras bestimmt werden.

$$
\begin{aligned}
s_{i,j} &= \sqrt{\Delta x^2 + \Delta y^2} \\
s_{i,j} &\overset{(31)}{=} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}
\end{aligned} \tag{32}
$$

6.5.2 Zusätzliches Längenstück

Da die Schneidewinkel λ bereits bestimmt wurden (siehe Tabelle in Kapitel 6.4) und die Breite b der Klötze festgelegt ist (hier $b = 4LE$), kann die zusätzliche Länge $l_{E,i}$ (i ist der Zählindex, der die aktuelle x-Stelle angibt, bei der die Extralänge $l_{E,i}$ auftritt)(in Abbildung 25 ist $l = l_{E,i}$) mit Hilfe des tan bestimmt werden.

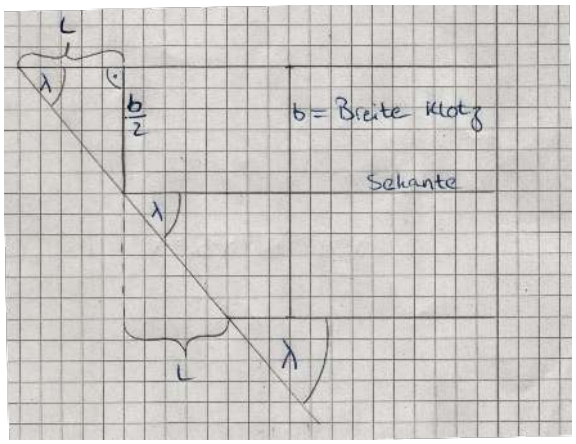


Abbildung 25: Länge an den Seiten der Klötze

$$
\begin{aligned}
\tan(\lambda) &= \frac{\frac{b}{2}}{l_{E,i}} \\
l_{E,i} &= \frac{b}{2 \cdot \tan(\lambda)}
\end{aligned} \tag{33}
$$

Zu beachten ist, dass jeder Klotz auf beiden Seiten um unterschiedliche Längen $l_{E,i}$ länger ist, da die Winkel an den x-Werten ebenfalls unterschiedlich von einander sind.

6.5.3 Länge Außen

Da nun die Sekantenlängen mit Hilfe von Formel (32) und die zusätzlichen Längen $l_{E,i}$ mit Hilfe von Formel (33) bestimmt wurden, können die finalen Klotzlängen bestimmt werden, indem außen die Sekantenlänge mit den zwei jeweiligen zusätzlichen Längenelementen addiert werden.

$Klotz_k$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$Laengeaußen_k in LE$	8.4	8.1	8.1	8.1	8.2	8.3	8.5	8.8	9.0	8.8	8.5	8.3	8.2	8.1	8.1	8.1	8.4

6.5.4 Länge Innen

Die innere Länge wird ähnlich wie die äußere bestimmt. Nur werden hier die zusätzlichen Längenelemente von der Sekantenlänge subtrahiert (siehe Abbildung 25).

$Klotz_k$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$Laengeinnen_k in LE$	7.6	7.9	7.9	7.9	7.8	7.7	7.4	7.0	6.7	7.0	7.4	7.7	7.8	7.9	7.9	7.9	7.6

Dadurch sind jetzt alle Abmaße der Klötze bestimmt und der Druck der Klötze kann erfolgen. Dieser ist im nächsten Unterkapitel beschrieben.

6.6 Druck der Klötze

Ursprünglich wollte ich die Klötze aus Holz bauen. Hierbei hätte ich die Außenlängen der Klötze und die Schneidewinkel benötigt. Doch da ich einen 3D-Drucker besitze und der Druck höchstwarscheinlich schneller, weniger fehleranfällig und deutlich genauer ist, habe ich mich dazu entschlossen, die Klötze mit diesem zu drucken.

Für den Druck benötigte ich nur die Eck-Koordinaten der Klötze. Diese kann ich mit Hilfe der Sekantenlänge und der zusätzlichen Längenstücke bestimmen (siehe Abbildung 26).

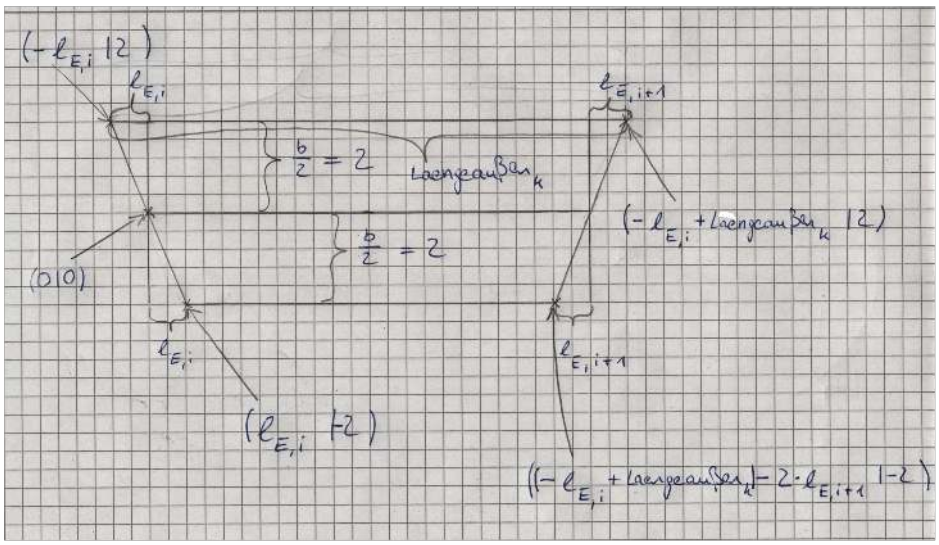


Abbildung 26: Eck-Koordinaten mit Punkt P_i als Nullpunkt

Der Punkt P_i wird als Nullpunkt gewählt. Dadurch wird es leichter, den Klotz zur Kontrolle an die richtige Position zu verschieben (siehe Kapitel 6.7).

Mit den Punkten der Ecken ist jetzt der Querschnitt eines Klotzes definiert. Im letzten Schritt habe ich daraus einen dreidimensionaler Körper erzeugt. Dabei wird der Querschnitt „nach oben gezogen“. Also erhalte ich acht Punkte, bei welchen je zwei Punkte übereinander liegen. Für die Höhe eines Klotzes habe ich hier $4LE$ gewählt.
Diese Punkte habe ich aus meinem Python-Programm als sogenannt stl-Datei exportiert (siehe Anhang) und gedruckt. Angaben zu den Koordinaten der Punkte befinden sich im Anhang.

Der Bogen aus Holzstücken war mit $1LE = 1cm$ geplant. Damit wäre der Bogen insgesamt $1,36m$ lang und $60cm$ breit geworden. Klötze für solch einen Bogen zu drucken, hätte jedoch relativ lang gedauert. Da der Druck genauer erfolgen kann, als das Schneiden per Hand, habe ich mich dazu entschieden, die Klötze auf die halbe Größe zu skalieren. Das Skalieren ist laut [Grothmann] erlaubt.

Nach dem ersten Druck war der Bogen jedoch noch realtiv instabil, da die Klötze genauso breit, wie hoch waren (siehe Abbildung 27).

Abbildung 27: Relativ instabiler Bogen von der Seite. Die Klötze sind nur $2cm$ hoch gedruckt, deshalb kippt der gesamte Bogen leicht nach rechts oder links.

Daher habe ich mich dafür entschieden die Höhe der Klötze nach Skalieren auf $3cm$ zu setzten. Dadurch sind die Klötze nur $2cm$ breit und $3cm$ hoch.
Der Druck der 17 Klötze mit diesen Maßen hat knapp sieben Stunden genauert.

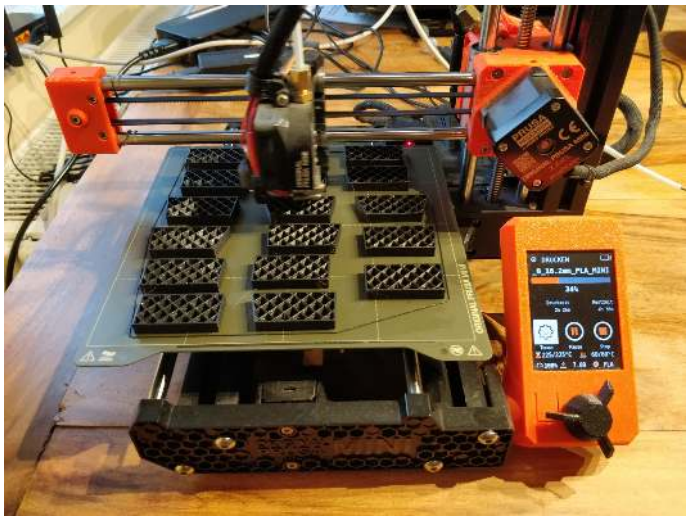


Abbildung 28: Laufender Druck der Klötze

Da der Drucker immer an den Ecken der Klötze ab und wieder ansetzt, haben sich dort Erhöhungen im Vergleich zur Fläche gebildet (siehe Abbildung 29).

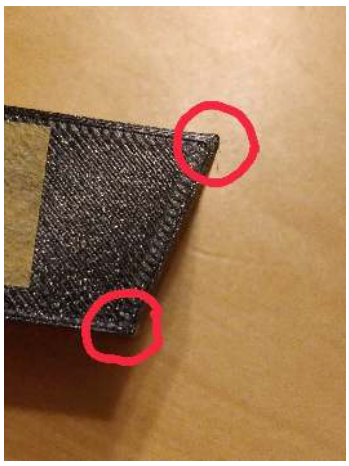


Abbildung 29: Erhöhung an den Ecken der Klötze

Dadurch ist die Fläche, mit welcher die Klötze aufeinander aufliegen extrem verkleinert. Um dies zu beheben, habe ich die Klötze an den Seiten abgeschliffen. So ist die Reibung größer und der Bogen stabiler.

Numerische Abmaße der einzelnen Klötze

$Klotz_k$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$Laengeinnen_k$ in cm	3.8	4.0	4.0	3.9	3.9	3.8	3.7	3.5	3.4	3.5	3.7	3.8	3.9	3.9	4.0	4.0	3.8
$Laengeaußen_k$ in cm	4.2	4.0	4.0	4.1	4.1	4.1	4.3	4.4	4.5	4.4	4.3	4.1	4.1	4.1	4.0	4.0	4.2

Der finale Bogen hat eine Länge von $68cm$, eine Breite am Boden von $30cm$, eine Klotzbreite von $2cm$ und eine Klotz Höhe von $3cm$.

Abbildung 30: Verschiedene Bilder des Bogens aus verschiedenen Aufbauten

Abbildung 31: Liegender Bogen vor dem Hinstellen

6.7 Verschiebung der Klötze mit Hilfe einer Drehmatrix

Bevor der Druck der Klötze erfolgte, wollte ich mich vergewissern, dass ich keinen Fehler bei deren Berechnung gemacht habe.
Um dies zu kontrollieren, habe ich die einzelen Klötze an die richige Stelle im Koordinatensystem verschoben und die Funktion mit den Klötzen in meinem Python-Programm zeichnen lassen (siehe Abbildung 32).

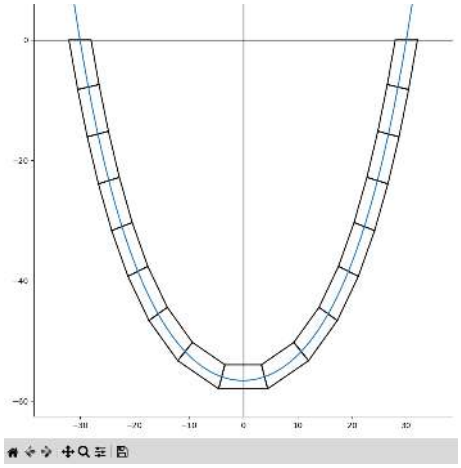


Abbildung 32: Klötze an der richtigen Stelle auf dem in blau gezeichneten Funktionsgraphen.

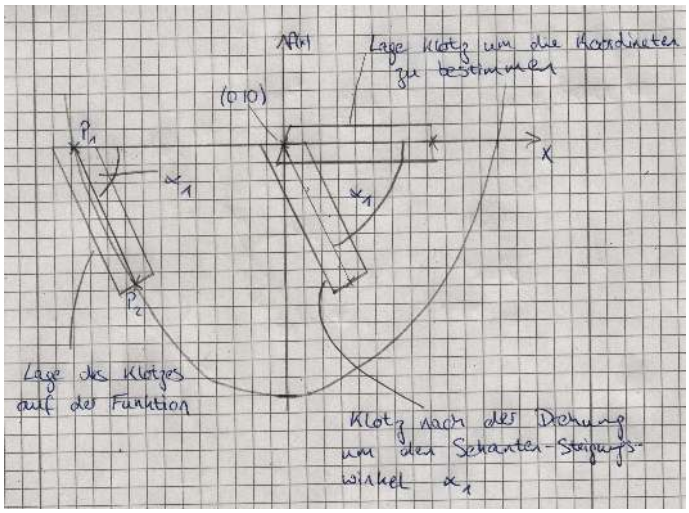


Abbildung 33: Drehung des Klotzes um den Sekatensteigungswinkel

Dafür habe ich die Punkte der einzelnen Klötze (siehe Abbildung 26) als Ortsvektoren betrachtet und um den Sekantensteigungswinkel des aktuellen Klotze gedreht (siehe Abbildung 33).

Da ich bei der Bestimmung der Koordinaten der Eckpunkte den Nullpunkt nicht auf eine Ecke gelegt habe, sondern auf die Hälfte einer Seite (siehe Abbildung 26), dort wo der Funktionsgraph den Klotz schneiden würde, ist das Verschieben ebenfalls unkompliziert. Denn der Nullpunkt der Klotzes muss dementsprechent nur auf den richtigen Punkt auf der Funktion geschoben werden. Und da die Koordinaten der Klotzpunkte bekannt sind (siehe Tabelle aus Kapitel 6.3), ist der Vektor der zum Verschieben benötigt wird direkt klar.
Jeder $Klotz_k$ hat vor dem Drehen und Verschieben die Punkte A_k, B_k, C_k, und D_k. Jetzt werden diese Punkte mit der Drehmatrix D multipliziert (siehe Grundlagen Formel (7)).

$$D = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

Und anschließend werden sie mit dem Ortsvektor $\vec{a_i}$ des aktuellen Punktes addiert. Daraus ergibt sich für die Punkt A'_k, B'_k, C'_k, und D'_k :

$$\begin{aligned} A'_k &= A_k \cdot D + \vec{a_i} \\ B'_k &= B_k \cdot D + \vec{a_i} \\ C'_k &= C_k \cdot D + \vec{a_i} \\ D'_k &= D_k \cdot D + \vec{a_i} \end{aligned}$$

Macht man diese Rechnung für jeden Klotz und lässt die daraus entstandenen Klötze zeichnen, erhält man Abbildung 32. Da der in Abbildung 32 gezeichnete Bogen korrekt ist, habe ich folglich keine Fehler bei der Berechnung der für den Bogen benötigten Werte gemacht, so dass ich die Klötze ohne Fehlversuch drucken konnte.

7 Anwendung in der Architektur

Die Kettenlinie spielt überwiegend im Bereich der Bogenkonstruktion und im Brückenbau eine Rolle.

Bogenkonsturktion

Das wohl offensichtlichste und bekannteste Beispiel für eine Anwendung der Kettenlinie im Bogenbau ist das Gateway Arch in Missouri, Vereinigte Staaten (siehe Abbildung 34).

Abbildung 34: Gateway Arch (Missouri, Vereinigte Staaten) Bildquelle: Wikipedia, Lizenz: CC-BA-SA 3.0
`https://upload.wikimedia.org/wikipedia/commons/thumb/b/b9/St_Louis_Gateway_Arch.jpg/465px-St_Louis_Gateway_Arch.jpg`

Die Konstruktion ist jedoch nur eine Annäherung an die Kettenlinie. Dieses korrekt zu realisieren war schierig, da das belasten der Kettenkurve mit einer Masse die Kettenkurve beeinflusst.

Verwendung von Hängemodellen

Hängemodelle sind an Ketten überkopf aufgehangene Modelle. Mit Hilfe dieser können Tragwerke von Gebäuden simuliert werden.

So wurde beispielsweise die Sagrada Família von Antonio Gaudí (1852 bis 1926) mit Hilfe von Hängemodellen geplant. Dafür hängte er Massestücke mit Fäden an eine Kette (siehe Abbildung 35).

Dreht man diese Hängemodelle um, ergeben die Ketten/Schnüre die Form eines stabilen Bogens für die darauf lastende Masse (diese Masse wurde an die Ketten/Schnüre dran gehangen um den Bogen zu simulieren). Die Bestimmung der Kettenkurven erfolgt jedoch bei Gaudí noch empirisch.
Vor der Entwicklung von Computerprogrammen zur Bogenkonstruktion war der Bau mit Hilfe von Hängemodellen eine gängige Methode.
Die Schilderung der Verwendung von Hängemodellen ist einer Publikation von [Graefe] entnommen.

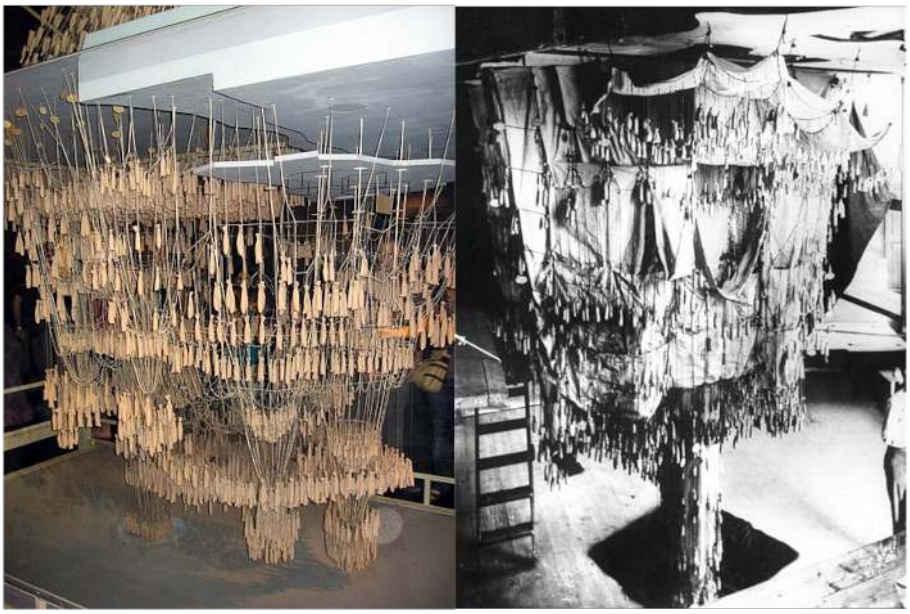


Abbildung 35: Hängemodelle der Sagrada Família; Bildquelle: Wikipedia, Lizenz: CC-BA-SA 3.0
`https://de.wikipedia.org/wiki/H%C3%A4ngemodell`

Brückenbau

Im Bau „normaler“Brücken wurden die tragenden Bögen ebenfalls mittels der Kettenlinie gebaut. Der Bau von Hängebrücken ist jedoch interesanter zu betrachten, da die tragenden Seile die Form einer Parabel bilden. Das liegt daran, dass sich eine Kettenlinie, an welche eine Masse homogen verteilt aufgehangen ist (wie beispielsweise bei einer Hängebrücke), zu einer Parabel ausformt (siehe [Hajiabadi, S. 10]). Also können die Seile einer Hängebrücke durch eine hängende Kette, auf welche eine Masse homogen aufgeteilt ist simuliert werden, (siehe Abbildung 36).

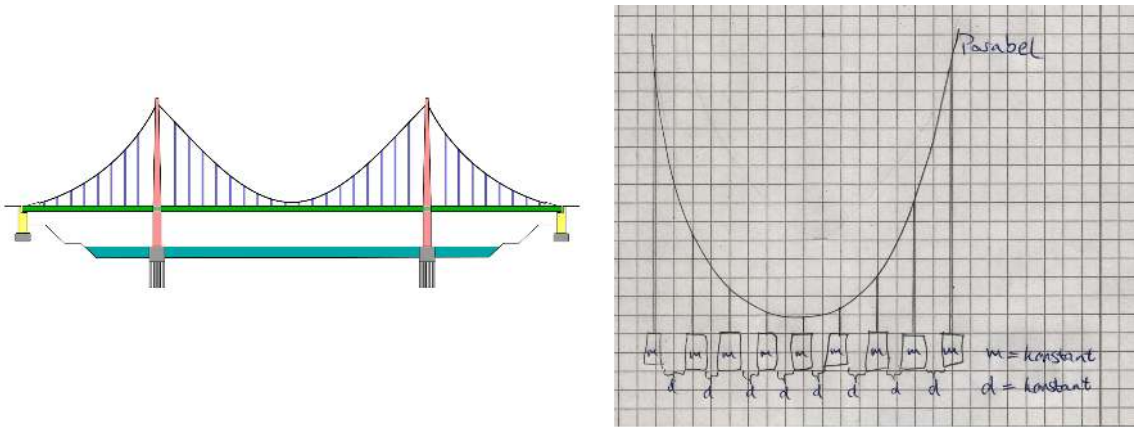


Abbildung 36: Grüne Fahrbahn wird durch die homogene Massenverteilung simuliert. Die Parabel simuliert das tragende Seil
Bildquelle: Wikipedia, Lizenz: CC-BA-SA 3.0 : `https://de.wikipedia.org/wiki/H%C3%A4ngebr%C3%BCcke#/media/Datei:H%C3%A4ngebr%C3%BCcke_Schema.svg`

Außerhalb der Architektur findet die Kettenlinie ebenfalls Anwendung in der Kunst und im Design. So sind beispielweise im McDonalds-Logo (siehe Abbildung 37) zwei Kettenlinien enthalten (siehe [Hajiabadi, S. 2]).

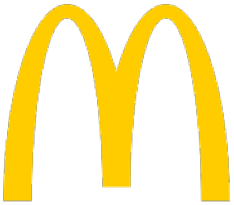


Abbildung 37: Zwei Kettenlinie im McDonalds-Logo. Bildquelle: Wikipedia, Lizenz: CC-BA-SA 3.0
`https://upload.wikimedia.org/wikipedia/commons/thumb/3/36/McDonald%27s_Golden_Arches.svg/400px-McDonald%27s_Golden_Arches.svg.png`

8 Fazit

Durch den Bau des frei stehenden Bogens habe ich nicht nur Einblicke in die theoretische Mathematik der Kettenlinie erhalten, sondern habe diese an einem praktischen Beispiel veranschaulicht und so mein eigenes Verständis vertieft.
Um den Fehler beim Bestimmen der numerischen Ergebnisse zu vermindern, schrieb ich ein Python Programm, welches mit den Parametern der Bogenlänge L, der Breite B und der Klotzbreite b (siehe Kapitel 6) alle für den Bau benötigten Werte bestimmt und automatisch die stl-Dateien der Klötze liefert. Diese werden benötigt, um die Klötze mit dem 3D-Drucker zu drucken.

Das Schreiben dieses Programmes war realtiv zeitaufwendig, da ich mich zuerst mit der Programmiersprache Python auseinander setzten musste, die ich bis dato nicht so gut beherrschte, dass ich ein solche Programm ohne Schwierigkeiten hätte schreiben können. Doch diese Zeitinvestition ermöglicht es mir, jetzt einen beliebigen Bogen ohne viel händische Rechnerei zu bauen oder zu simulieren, da das Programm auch die Möglichkeit bietet den Bogen zu zeichnen.

Im Zuge der Facharbeit habe ich mich in das professionelle Setztprogramm LaTeX eingearbeitet. Dies war zwar ebenfalls relativ aufwendig, doch da dieses Programm gängig für wissenschaftlich Arbeiten ist, kann ich meine Kenntnisse gut in einem potenziellen Studium anwenden.

9 Literaturverzeichnis und Eigenständigkeitserklärung

Literatur

[Athen/Bruhn] Athen/Bruhn: Lexikon der Schul-Mathematik, Teil 1, Weltbild Verlag, Augsburg 1994

[Fischer] Fischer, Gerd: Lineare Algebra, Vieweg, Braunschweig 1986

[Formelsammlung] Sieber, H.: Mathematische Formeln. Erweiterte Ausgabe E, Klett Schulbuchverlag, Stuttgart 1994

[Glück] Glück, Bernd: Die Kettenlinie in der Mechanik und ihr Pendant bei der Wärmeleitung in Rippen, 2022, `https://berndglueck.de/dl/?dl=Waermeuebertragung+Kettenlinie-Rippentemperatur.pdf`, abgerufen am 24.02.2024

[Graefe] Graefe, Rainer: Zum Entwerfen mit Hilfe von Hängemodellen, Zeitschrift Werk, Bauen + Wohnen, Band 70, Heft 11, 1983, `https://doi.org/10.5169/seals-53542`, abgerufen am 24.02.2024

[Grothmann] Grothmann, R.: Die Kettenlinie. Geometische Eigenschaften der Kettenkurve, http://mathsrv.ku-eichstaett.de/MGF/homes/grothmann/Projekte/Kettenlinie/geometrie.html, abgerufen am 24.02.2024

[Hajiabadi] Hajiabadi, Maikel: Die Kettenlinie, Univeristät Mainz, Wintersemester 2016/17, `https://download.uni-mainz.de/mathematik/Algebraische%20Geometrie/Lehre/Sem-Ausgewaehlte-hoehere-Kurven-WS2016-17/Hajiabadi%20Maikel_Die%20Kettenlinie_090217.pdf`, abgerufen am 24.02.2024

[Lern-Helfer] Lern-Helfer: Die Kettenlinie, 2010, `https://www.lernhelfer.de/schuelerlexikon/mathematik-abitur/artikel/die-kettenlinie#`, abgerufen am 24.02.2024

[Müller-Fonfara] Müller-Fonfara, Robert: Mathematik.verständlich, Bassermann, München 2004

[Stifel] Stifel, Michael: `https://mathepedia.de/Kettenlinie.html`, abgerufen am 24.02.2024

Eigenständigkeitserklärung

Ich erkläre, dass ich die Facharbeit ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benutzt habe.

10 Anhang

10.1 Grundlagen: Matrix Vektor Rechnung

Die für die Drehmatrix (Kapitel 3) verwendeten Grundlagen werden hier behandelt.

Vektoren sind mathematische Objekte, die sowohl eine Größe als auch einen Richtung haben.
(Ein Vektor ist eine Klasse von paralleler, gleichlanger, gleichgerichteter Pfeile.)
Matrizen sind zweidimensionale Anordnungen von Zahlen in Form einer Tabelle.

10.1.1 Graphische Darstellung Vektoren

Vektoren können als Pfeile in einem Koordiatensystem visualisiert werden. Hierbei ist die Länge des Pfeils die Größe des Vektors. Die Richtung des Vektors wird aus den einzelnen Komponenten ersichtlich.
Für einen zweidimensionalen Vektor $\vec{a} \epsilon \mathbb{R}^2$, schreibt man:

$$\vec{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

Wobei a_1 die Entfernung parallel zur x-Achse ist und a_2 die Entfernung parallel zur y-Achse.
Der Vektor $\vec{a} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$ ist in Abbildung 38 visualisiert.

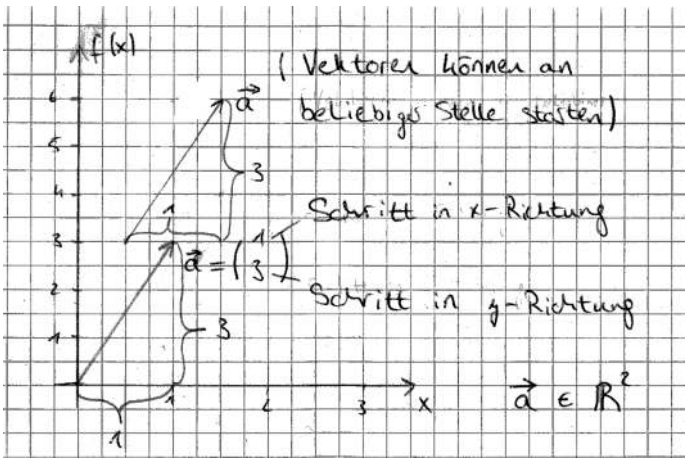


Abbildung 38: Vektoren können parallel beliebig verschoben werden. Dargestellt sind zwei Pfeile, die zum selben Vektor $\vec{a}$ gehören.

Für einen dreidimensionalen Vektor $\vec{a} \epsilon \mathbb{R}^3$, schreibt man:

$$\vec{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Wobei a_1 die Entfernung parallel zur x-Achse ist, a_2 die Entfernung parallel zur y-Achse und a_3 die Entfernung parallel zur z-Achse ist.

Der Vektor $\vec{a} = \begin{pmatrix} 2 \\ 3 \\ -1 \end{pmatrix}$ ist in Abbildung 39 visualisiert.

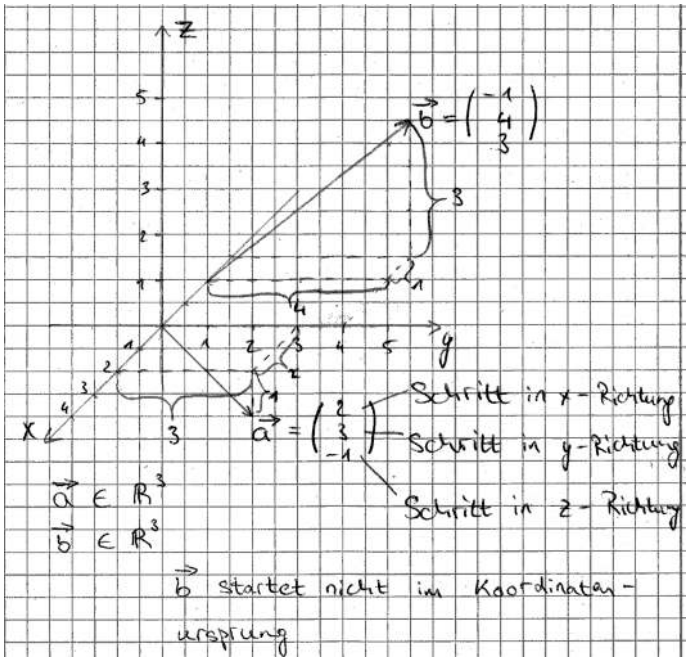


Abbildung 39: Zwei verschiedene Vektoren im dreidimensionalen Raum

10.1.2 Betrag eines Vektors

Der Betrag eines Vektors $\vec{a}$ wird mit dem gleichen Variablennamen a, ohne Vektorpfeil, bezeichnet.

$$a := |\vec{a}|$$

Der Betrag wird nach dem Satz von Pythagoras bestimmt (siehe Abbildung 39).

$$a \quad = \quad |\vec{a}| = \sqrt{a_1^2 + a_2^2 + a_3^2}$$

10.1.3 Addition von Vektoren

Die Addition von Vektoren wird in Kapitel 6.7 angewant, um Punkte in einem Koordinatensystem zu verschieben. Weshalb dies geht, und wie Vektoren addiert werden, wird hier behandelt.
Vektoren werden addiert, indem die einzelnen Komponenten addiert werden.

$$\vec{a} + \vec{b} := \begin{pmatrix} a_1 + b_1 \\ a_2 + b_2 \end{pmatrix}$$

Die Addition von Vektoren ist kommutativ.

10.1.4 Matrix-Vektor-Multiplikation

Durch bestimmte Matrizen (Drehmatrizen) können Vektoren im Koordinatensystem um den Koordinatenursprung gedreht werden. Diese Drehung wird durch eine Matrix-Vektor-Multiplikation beschrieben.
Der Vektor

$$\vec{b} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

soll mit der Matrix

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

multipliziert werden. Die Multiplikation einer Matrix mit einem Vektor erfolgt durch die Multiplikation von Zeile mal Spalte.

$$A \cdot \vec{b} := \begin{pmatrix} a_{11} \cdot b_1 + a_{12} \cdot b_2 \\ a_{21} \cdot b_1 + a_{22} \cdot b_2 \end{pmatrix}$$

Oder allgemein geschrieben als:

$$A \cdot \vec{b} = \begin{pmatrix} \sum_{i=1}^{n} a_{1i} \cdot b_i \\ \sum_{i=1}^{n} a_{2i} \cdot b_i \end{pmatrix} \tag{34}$$

Hier $n = 2$, da es sich um eine 2×2 Matrix handelt.

10.2 Numerische Werte

Faktoren der Funktionsgleichung

Die allgemeine Funktionsgleichung für eine Kettenlinie (22) lautet:

$$f(x) = a \cosh\left(\frac{1}{a}x\right) - y_0$$

Die Faktoren a und y_0 für die Bogen beschreibende Funktion lauten in meinem Fall:

$$a \approx 12.541937507689 \qquad y_0 \approx -69.14694637731643$$

Diese Werte habe ich mittels meines Pythonprogrammes bestimmt.
Die Bogen beschreibende Funktion lautet also:

$$f(x) = 12.541937507689 \cosh\left(\frac{1}{12.541937507689}x\right) - 69.14694637731643$$

Python Programm Dokumentation

March 7, 2024

Der Code wird im Folgenden erst beschrieben und unter der jeweiligen Beschreibung eingebunden.

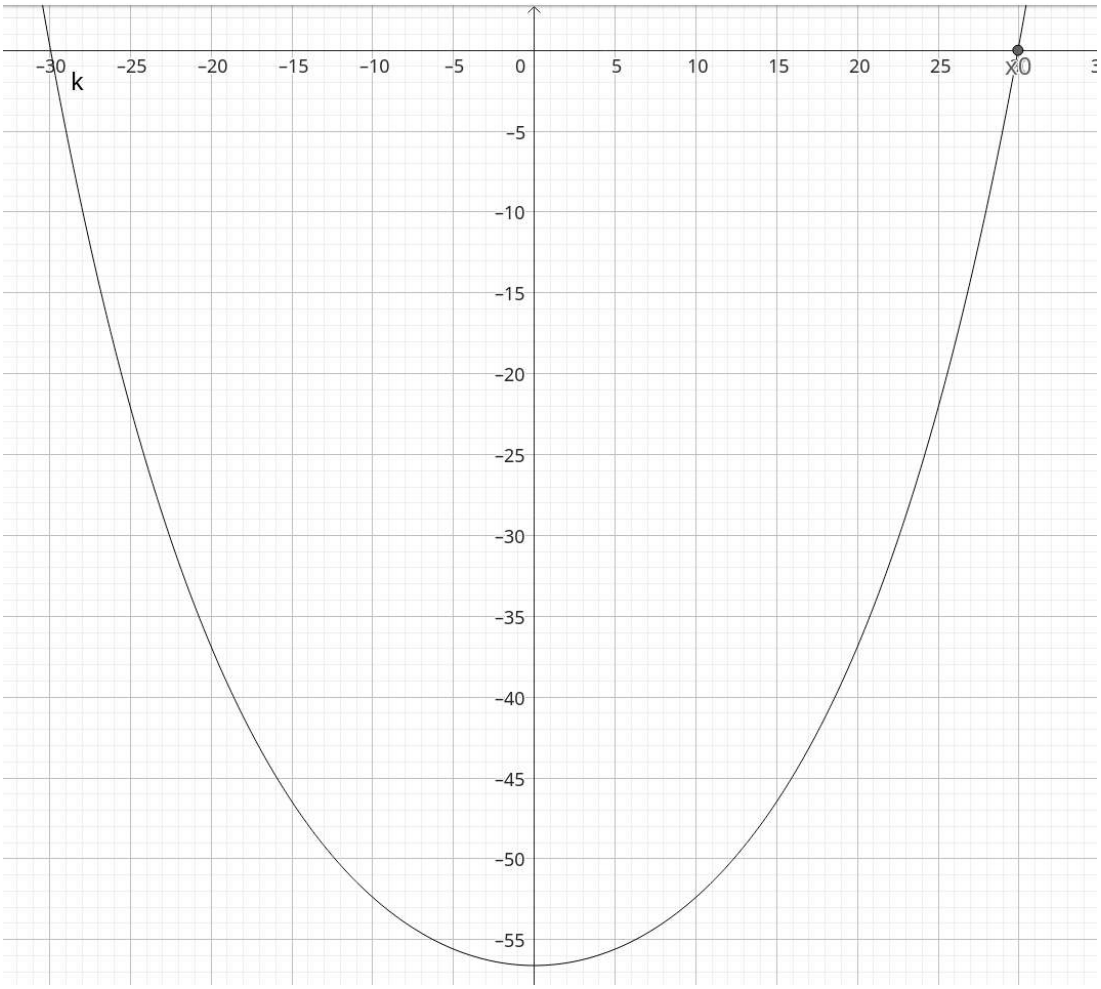
1 Python Porgramm frei stehender Bogen

Zu Beginn werden die für das Programm benötigten Python Bibliotheken importiert.

[1]:
```
import math
import numpy as np
import matplotlib.pyplot as plt
from shapely.geometry import Polygon
from stl import mesh
```

1.1 Bestimmung der Funktionsgleichung

Für den Bau des Bogens, hab ich bestimmte Parameter festgelegt, die der Bogen erfüllen soll. Einerseits die Länge L des Bogens (Bogenlänge zwischen den Nullstellen) und die Nullstelle x_0, bzw. die Hälfte der Bogenbreite (siehe Abbildung). Zu dem ist die Anzahl an Klötzen, die Breite eines Klotzes und die durch einen Klotz beschriebene Bogenlänge determiniert. Der Bogen wird durch eine hängende Kette beschrieben.



Hier werden die Parameter des Bogens initialisiert. Die Bogenlänge L hat hier den Variablennamen L. Die Nullstelle x_0 den Namen x0.

[2]:
```
L = 136
x0 = 30
anzahlKloetze = 17
laengeDurchKlotzSimuliert = 8 #die Bogenlänge, die durch einen Klotz simuliert␣
↪wird
breiteKloetze = 4
```

---

2 Ermittlung Faktor a als Nullstelle von $g(a)$ mittels Intervallhalbierung durch eine rekursive Python-Funktion

Für die Funktion, welche die Kette beschreibt, gilt: $f(x) = a \cdot \cosh\left(\frac{1}{a}x\right) + y_0$ (siehe PDF, Kapitel 5)

Der Faktor a ist die Nullstelle der Funktion $g(a) := L - 2a \cdot \sinh\left(\frac{1}{a}x_0\right)$ (siehe PDF, Kapitel 6.2) Die numerische Bestimmung **der Nullstelle** erfolgt hier mit Hilfe von Intervallhalbierung:

Beim Funktionsaufruf, wird eine untere und eine obere Grenze festgelegt. Für die Variable a wird die Mitte zwischen diesen Werten gewählt. Danach wird in der Variable f_a der aktuelle Wert von $g(a) = L - 2a \cdot \sinh\left(\frac{1}{a}x\right)$ abgelegt.

Ist dieser Wert kleiner als $\frac{1}{10000000}$, also näherungsweise Null, wird der Wert von a zurückgegeben.

Andernfalls wird geprüft ob der Wert der Variablenwert f_a größer oder kleiner ist als Null. Dann folgt ein **rekursiver** Funktionsaufruf, ist f_a größer Null wird der Variablenwert von a als obere Grenze übergeben und die untere Grenze bleibt gleich. Ist f_a kleiner erfolgt der **rekursive** Aufruf mit dem Variablenwert von a als untere Grenze, die obere Grenze bleibt gleich.

[3]:
```
def a_bestimmen(unterGrenze,oberGrenze):
    a = (unterGrenze+oberGrenze)/2
    f_a = (L-2*a*math.sinh(1/a*x0))
    if( abs(f_a) < 1/10000000 ):
        return a
    elif (f_a < 0):
            return a_bestimmen(a,oberGrenze);
    else:
        return a_bestimmen(unterGrenze,a);
```

[4]:
```
a = a_bestimmen(10,16)
print("a : ", a)
```

```
a :  12.541937507689
```

Setzt man $f(x_0) = 0$ und formt nach y_0 um erhält man: $y_0 = -a \cdot \cosh\left(\frac{1}{a}x\right)$ (siehe Kapitel 6.2) Dieser Wert wird in der Variablen y0 abgespeichert.

[5]:
```
y0 = -a*math.cosh(1/a*x0)
```

Jetzt sind alle Größe der Funktion $f(x) = a \cdot \cosh\left(\frac{1}{a}x\right) + y_0$ bekannt und die Funktion der Kettenlinie und deren Ableitung werden hier initialisiert, damit in anderen Funktionen der Funktionswert nicht händisch über die Funktionsgleichung bestimmt werden muss, sondern einfach kettenlinie(x) benutzt werden kann.

[6]:
```
kettenlinie = lambda x: (a*np.cosh(1/a * x)+y0)
ableitung = lambda x: (np.sinh(1/a * x)) #siehe Ableitungsregeln PDF
```

2.1 Bestimmung der x und y-Werte

Die Funktion $x(l) = a \cdot arcsinh\left(\frac{l}{a}\right)$ (siehe PDF Kapitel 6.3) liefert einen x-Wert in Abhängigkeit von einer Bogenlänge l. Hier liefert die Funktion x_werte_l(l) den x-Wert zur zugehörigen Bogenlänge l.

[8]:
```
x_werte_l = lambda l: (np.arcsinh(l/a) * a)
```

Um eine Liste der benötigten x-Werte zu erhalten, läuft in der folgenden Funktion eine for Schleife von -(der Hälfte der vorgegebenen Bogenlänge (Nullstelle im negativen Bereich)) bis zur +(Hälfte der vorgegeben Bogenlänge (Nullstelle auf positiver Seite)). In jedem Schleifendurchlauf, wird i um die größe der Variable laengeDurchHolzStueckeSimuliert (Länge vom Bogen, welche durch einen Klotz beschrieben wird) erhöht, und mit diesem i wird die Funktion x_werte_l(l) aufgerufen. i ist also praktisch die momentane Bogenlänge. Der Rückgabewert wird zur Liste hinzugefügt.

[21]:
```
def x_werte_holzstuecke():
    x_werte_holzstuecke = [] #die Liste wird initialisiert
    for i in range (-int(L/2), int(L/2+1), laengeDurchKlotzSimuliert):␣
 ↪#for-Schleife die über die Bogenlänge iteriert
        x_werte_holzstuecke.append(x_werte_l(i)) #die Liste wird um den␣
 ↪Rückgabewert der Funktion x_werte_l(i) erweitert
    return x_werte_holzstuecke #die Liste wird zurückgegeben
```

[22]:
```
x_werte = x_werte_holzstuecke() #in der Variable x_werte wird die Liste der␣
 ↪x-Werte gespeichert
print("x Werte:")
print(x_werte)
```

```
x Werte:
[-30.000000004186834, -28.459695195406937, -26.708667641693076,
-24.682268897417455, -22.282466596571965, -19.352603653310144,
-15.629065100279782, -10.666755857440647, -3.9351171525557147,
3.9351171525557147, 10.666755857440647, 15.629065100279782, 19.352603653310144,
22.282466596571965, 24.682268897417455, 26.708667641693076, 28.459695195406937,
30.000000004186834]
```

Um eine Liste mit den zugehörigen y-Werte zu bekommen, läuft die folgende Funktion durch die Liste der x-Werte, welche als Parameter übergeben wird, und fügt den jeweiligen Funktionswert von kettenlinie = lambda x: (a*np.cosh(1/a* x)+y0) zur Liste hinzu.

[11]:
```
def y_werte_holzstuecke(xWerteListe):
    yWerte = []
    for i in xWerteListe:
        yWerte.append(kettenlinie(i))
    return yWerte
```

[12]:
```
y_werte = y_werte_holzstuecke(x_werte)
print("y Werte:")
print(y_werte)
```

```
y Werte:
[2.2700234580952383e-08, -7.8501259341093075, -15.655823782564553,
```

```
-23.39434867569104, -31.024771774511088, -38.466330404901925,
-45.53973992575219, -51.78894544268181, -55.982593128987965,
-55.982593128987965, -51.78894544268181, -45.53973992575219,
-38.466330404901925, -31.024771774511088, -23.39434867569104,
-15.655823782564553, -7.8501259341093075, 2.2700234580952383e-08]
```

2.2 Bestimmung der Winkel zwischen den Klötzen

Um die Winkel zwischen den Klötzen zu bestimmen, werden die Sekanten-Steigungswinkel der Klötze benötigt. Die folgende Funktion bestimmt den Steigungswinkel der Sekante durch die Punkte p1 und p2:

[23]:
```
def sekantenSteigungsWinkel(p1, p2): #p1 und p2 werden als Parameter übergeben
    m = (kettenlinie(p1)-kettenlinie(p2))/(p1-p2) #Steigung wird bestimmt
    return (np.arctan(m)/(2 * math.pi) * 360) #bestimmung des Steigungswinkels␣
↪und umrechnen von Radiant in Grad
```

Die folgende Funktion liefert eine Liste mit den Sekantensteiungswinkeln, welche jeweils durch einen Klotz laufen. Dafür wird erst der erste Steiungswinkel hinzugefügt und zu letzt der letzte (siehe PDF Kapitel 6.4). Dazwischen läuft eine for-Schleife durch die Liste der x-Werte und ruf jeweils die Funktion sekantenSteigungsWinkel(x_werte[i], x_werte[i+1]) mit dem momentanen und dem darauf folgenden x-Wert auf.

[24]:
```
def sekantenSteigungsWinkelListe(x_werte):
    steigungsWinkel = []
    steigungsWinkel.append( np.arctan(␣
↪(kettenlinie(x_werte[0])-(-1)*kettenlinie(x_werte[1]))/
↪(x_werte[0]-x_werte[1]) ) / (2*math.pi)*360 ) # siehe PDF Besonderheiten␣
↪erster und letzter Klotz
    for i in range(0,len(x_werte)-1): # von 0 bis len-1
        steigungsWinkel.append(sekantenSteigungsWinkel(x_werte[i],␣
↪x_werte[i+1]))

    steigungsWinkel.append( np.arctan(␣
↪(kettenlinie(x_werte[len(x_werte)-1])-(-1)*kettenlinie(x_werte[len(x_werte)-2]))/
↪(x_werte[len(x_werte)-1]-x_werte[len(x_werte)-2]) ) / (2*math.pi)*360 ) ##
    return steigungsWinkel
```

[64]:
```
sekantenSteigungswinkelWerteListe = sekantenSteigungsWinkelListe(x_werte)
print("Sekanten-Steigungswinkel:")
print(sekantenSteigungswinkelWerteListe)
```

```
Sekanten-Steigungswinkel:
[78.898795820709, -78.898795883317, -77.35633239230725, -75.32608580825179,
-72.54140437136391, -68.50963158917503, -62.2370737375766, -51.54795029298983,
-31.92190536112552, -0.0, 31.92190536112552, 51.54795029298983,
62.2370737375766, 68.50963158917503, 72.54140437136391, 75.32608580825179,
```

```
77.35633239230725, 78.898795883317, -78.898795820709]
```

Mit Hilfe dieser Sekanten-Steigungswinkel wird nun der "Schneidewinkel", bzw. die Hälfte des Winkels zwischen den Klötzen (siehe PDF Kapitel 6.4), bestimmt. Dafür läuft die nachfolgende Funktion durch die Liste der Sekanten-Steigungswinkel und bestimmt nach Gleichung (30) (siehe PDF Kapitel 6.4) den Winkel λ (Hälfte des Winkels zwischen den Klötzen).

[30]:
```
def winkelZwischenKloetzenHalbe(sekantenSteigungsWinkel):
    listeWinkel = []
    listeWinkel.append(sekantenSteigungsWinkel[0])
    for i in range(1,len(sekantenSteigungsWinkel)-2):
        listeWinkel.append((180 - sekantenSteigungsWinkel[i+1] +␣
↪sekantenSteigungsWinkel[i])/2)
    listeWinkel.
↪append(abs(sekantenSteigungsWinkel[len(sekantenSteigungsWinkel)-1]))
    return listeWinkel
```

[37]:
```
winkelZwischenKloetzenHalbeWerte =␣
↪winkelZwischenKloetzenHalbe(sekantenSteigungswinkelWerteListe)
print("Winkel zwischen den Klötzen halbe (Schneidewinkel):")
print(winkelZwischenKloetzenHalbeWerte)
```

```
Winkel zwischen den Klötzen halbe (Schneidewinkel):
[78.898795820709, 89.22876825449512, 88.98487670797226, 88.60765928155607,
87.98411360890556, 86.86372107420078, 84.65543827770662, 80.18697753406784,
74.03904731943724, 74.03904731943724, 80.18697753406784, 84.65543827770662,
86.86372107420078, 87.98411360890556, 88.60765928155607, 88.98487670797226,
89.22876825449512, 78.898795820709]
```

2.3 Bestimmung der Klotzmaße

Zu Beginn wird die Länge der Sekanten, welche die Klötze beschreiben, bestimmt. Dies erfolgt, indem die Funktion mittels einer for-Schleife durch die Liste der x-Werte läuft und immer den aktuellen und den folgenden x-Wert nimmt und zu diesen die y-Werte bestimmt. Mit Hilfe des Satz von Pythagoras wird die Länge der Sekanten bestimmt. (siehe PDF Kapitel 6.5.1)

[33]:
```
def sekantenLaengen(xWerteListe):
    sekantenLaengen = []
    for i in range(0, len(xWerteListe)-1):
        y1 = kettenlinie(xWerteListe[i])
        y2 = kettenlinie(xWerteListe[i+1])
        sekantenLaengen.append(math.sqrt((xWerteListe[i+1]-xWerteListe[i])**2 +␣
↪(y2-y1)**2))
    return sekantenLaengen
```

[68]:
```
sekantenLaengenWerte = sekantenLaengen(x_werte)
print("Sekantenlängen:")
```

```
print(sekantenLaengenWerte)
```

```
Sekantenlängen:
[7.99981352543454, 7.999688518638961, 7.999441192504645, 7.998900408815565,
7.997555358723262, 7.993613801376608, 7.97979214105169, 7.931055451197266,
7.870234305111429, 7.931055451197266, 7.97979214105169, 7.993613801376608,
7.997555358723262, 7.998900408815565, 7.999441192504645, 7.999688518638961,
7.99981352543454]
```

Da die Sekanten durch die Mitte der Klötze läuft, kommen durch die Schräge beim schneiden auf der Außenseite des Bogens noch Längen dazu und innerhalb fallen genau diese weg (siehe PDF Kapitel 6.5.2). Die zusetzlichen Längen werden bestimmt, indem die Hälfte der Breite mit dem Tangens des jeweiligen Winkels im Klotz multipliziert wird, (siehe PDF Gleichung (33)).

[48]:
```
def zusetzlicheLaengenstuecke(winkelInHolzStueck):
    langeSeite = []
    for i in range(0,len(winkelInHolzStueck)):
        langeSeite.append(2 / math.tan(math.radians(winkelInHolzStueck[i])))
    return langeSeite
```

[76]:
```
zusetzlicheLaengenstueckeWerte =␣
↪zusetzlicheLaengenstuecke(winkelZwischenKloetzenHalbeWerte) #als Parameter␣
↪wird winkelZwischenKloetzenHalbeWerte übergeben, da diese Winkel jenen in␣
↪den Klötzen sind
print("Zusetzliche Längenelemente:")
print(zusetzlicheLaengenstueckeWerte)
```

```
Zusetzliche Längenelemente:
[0.39242805176953294, 0.026922692534157827, 0.03543819562185135,
0.04861142898944077, 0.07039676027437022, 0.10958625970258735,
0.18710338770379092, 0.3459281360982313, 0.5720159803737838, 0.5720159803737838,
0.3459281360982313, 0.18710338770379092, 0.10958625970258735,
0.07039676027437022, 0.04861142898944077, 0.03543819562185135,
0.026922692534157827, 0.39242805176953294]
```

Da nun alle Größen bekannt sind, kann die außere Länge der Klotze bestimmt werden. Da der erste und letzte Klotz von den anderen unterscheiden, liefert die folgende Methode nur die Mittleren Längen. Es wird die jeweilige Sekantenlänge mit den beiden zusetzlichen Länge addiert. Zu beachten ist, dass die beiden zusatzlängen Längen in einem Klotz verschieden voneinander sind.

[50]:
```
def laengeKloetzeAussenListe(sekantenLaengenListe, zusetzlicheLaengenListe):
    laengen = []
    for i in range(0,len(sekantenLaengenListe)):#1 bis len-1
        laengen.append(sekantenLaengenListe[i] + zusetzlicheLaengenListe[i] +␣
↪zusetzlicheLaengenListe[i+1])
    return laengen
```

[62]:
```
laengeKloetzeAussenWerteListe = laengeKloetzeAussenListe(sekantenLaengenWerte,␣
↪zusetzlicheLaengenstueckeWerte)
print("Klotzelängen Außen:")
print(laengeKloetzeAussenWerteListe)
```

```
Klotzelängen Außen:
[8.41916426973823, 8.062049406794971, 8.083490817115937, 8.117908598079376,
8.177538378700218, 8.290303448782986, 8.512823664853713, 8.848999567669281,
9.014266265858998, 8.848999567669281, 8.512823664853713, 8.290303448782986,
8.17753837870022, 8.117908598079376, 8.083490817115939, 8.06204940679497,
8.41916426973823]
```

Das gleiche geschieht in der nachfolgenden Funktion für die innere Länge. Hier werden die zusetzlichen Längenelemente von der Sekantenlänge subtrahiert.

[52]:
```
def laengeKloetzeInnenListe(sekantenLaengenListe, zusetzlicheLaengenListe):
    laengen = []
    for i in range(0,len(sekantenLaengenListe)):#1 bis len-1
        laengen.append(sekantenLaengenListe[i] - zusetzlicheLaengenListe[i] -␣
↪zusetzlicheLaengenListe[i+1])
    return laengen
```

[54]:
```
laengeKloetzeInnenWerteListe = laengeKloetzeInnenListe(sekantenLaengenWerte,␣
↪zusetzlicheLaengenstueckeWerte)
print("Klotzelängen Innen:")
print(laengeKloetzeInnenWerteListe)
```

```
Klotzelängen Innen:
[7.580462781130849, 7.937327630482952, 7.915391567893353, 7.8798922195517545,
7.817572338746304, 7.696924153970229, 7.446760617249668, 7.013111334725251,
6.7262023443638626, 7.013111334725251, 7.446760617249668, 7.696924153970229,
7.817572338746304, 7.8798922195517545, 7.915391567893353, 7.937327630482952,
7.580462781130849]
```

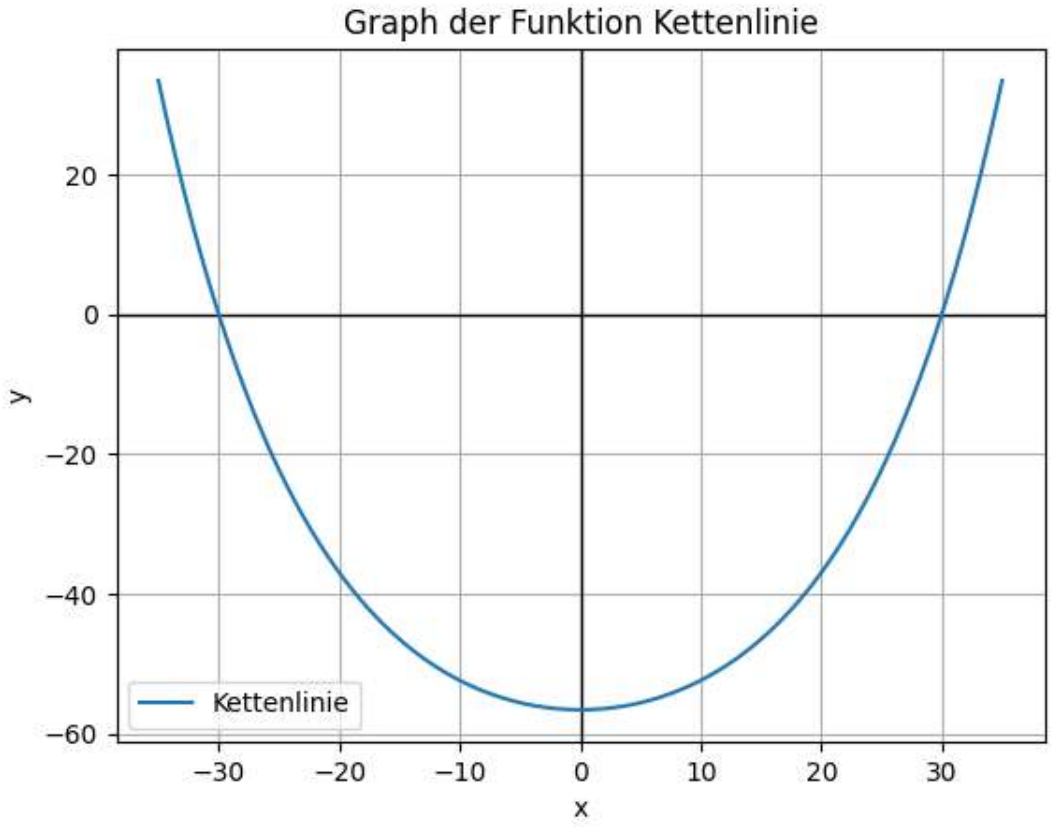
2.4 Zeichnen der Kettenlinie

Die folgende Funktion zeichnet eine Funktion f, die als Parameter übergeben wird. Hier ist diese Funktion nützlich, da man sich so ein Bild von der Kettenlinie machen kann.

[56]:
```
def funktionZeichnen(f, xmin, xmax, label=None):
    x_values = np.linspace(xmin, xmax, 300) #eine Liste an x-Werten wird␣
↪erzeugt. 300 x-Werte im Bereich zwischen xmin und xmax
    y_values = f(x_values) #die zugehörigen y-Werte werden bestimmt

    plt.axhline(0, color='black',linewidth=1)
    plt.axvline(0, color='black',linewidth=1)
```

```
    plt.plot(x_values, y_values, label=label) #die Punkte, die die x- und␣
↪y-Werte bilden werden eingezeichnet, da es relativ viele sind, sieht es nach␣
↪einer Kurve aus

    plt.xlabel('x') #Achsenbeschriftung
    plt.ylabel('y')
    plt.title(f'Graph der Funktion {label}')
    plt.grid(True)
    plt.legend()
    plt.show()
```

[59]:
```
funktionZeichnen(kettenlinie,-35,35, "Kettenlinie") #Funktionsaufruf mit der␣
↪Gleichung der zuvor bestimmten Kettenlinie
```



2.5 Speichern der Klötze und graphische Kontrolle

Die nachfolgende Funktion, speichert die Klötze als stl-Datei ab und zeichnet die Klötze zur Kontrolle. Da diese Funktion realativ komplex ist, sind einzelne Befehle erklärt. Die Erklärungen sind

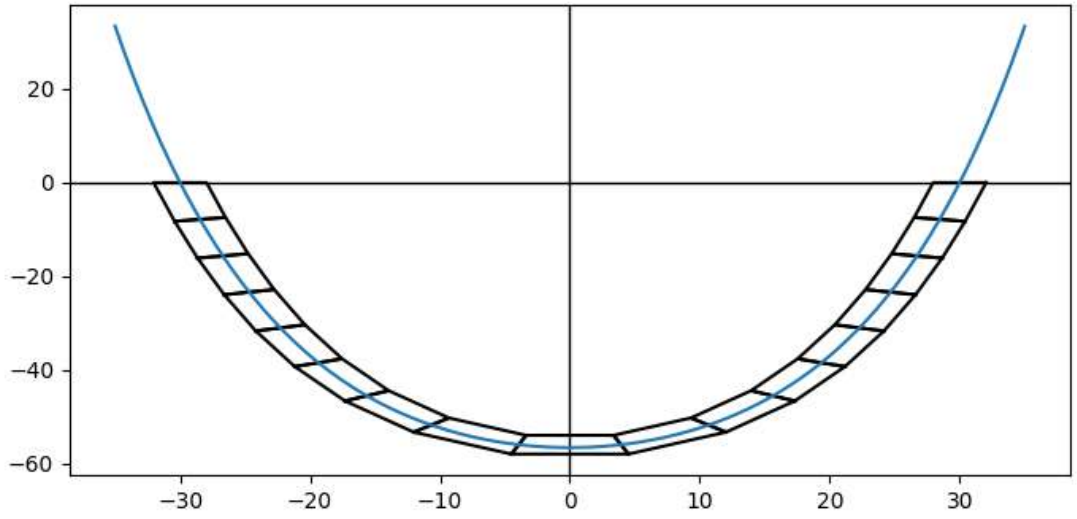
direkt hinter den Zeilen.

[81]:
```
def kloetzeAbspeichern(sekantenLaengen, laengeKloetzeAussen, zusetzlicheLaenge,␣
↪x_werte, y_werte, sekantenSteigungswinkelListe):
    for n in range (0, len(laengeKloetzeAussen)): #Schleife läuft von Null bis␣
↪zur Anzahl der Klötze
        theta = np.radians(sekantenSteigungswinkelListe[n+1]) + np.radians(180)␣
↪#theta ist der Winkel, um den der Klotz gedreht wird, bei n+1 weil trick mit␣
↪erster und letzter klotz +180 da rechtsrum gedreht wird
        c, s = np.cos(theta), np.sin(theta) #in c wird der cosinus von theta␣
↪geschrieben, in s der sin
        R = np.array(((c, -s), (s, c))) #Drehmatrix R wird initialisiert (siehe␣
↪PDF Gleichung (7))

        #die Koordinaten der Punkte werden bestimmt (siehe PDF Abbildung 26).␣
↪Hier liegt der Klotz noch an dem Nullpunkt
        a = np.array([-zusetzlicheLaenge[n+1], 2])
        b = np.array([ zusetzlicheLaenge[n+1],-2])
        c = np.array([-zusetzlicheLaenge[n+1]+laengeKloetzeAussen[n], 2])
        d = np.
↪array([-zusetzlicheLaenge[n+1]+laengeKloetzeAussen[n]-2*zusetzlicheLaenge[n],-2])

        v = np.array([x_werte[n+1], y_werte[n+1]]) #v ist der Ortsvektor des␣
↪Punktes, der beim bestimmen der Koordinaten auf den Nullpunkt gelegt wurde

        #die die Punkte a,b,c,d werden mit der Drehmatrix R multipliziert und␣
↪mit dem Vektor v addiert und liegen jetzt auf der richtigen Stelle auf dem␣
↪Funktionsgraphen
        a1 = np.dot(R,a)+v
        b1 = np.dot(R,b)+v
        c1 = np.dot(R,c)+v
        d1 = np.dot(R,d)+v

        plt.rcParams["figure.figsize"] = [7.00, 3.50] #legt die Größe der␣
↪Matplotlib-Figuren fest (Größen in Zoll)
        plt.rcParams["figure.autolayout"] = True #aktivieren das automatische␣
↪Layout

        polygon2 = Polygon([a1,c1,d1,b1,]) #eine Figur des Klotzes wird␣
↪erstellt. Hierbei sind die Seiten a1c1 c1d1 d1b1 und b1a1

        x, y = polygon2.exterior.xy #der Variable x werden die x-Werte der␣
↪Koordinaten zugeordnet, der Variable y die y-Werte
        plt.plot(x, y, c="black") #Die Punkte werden gezeichnet und gezeichnet␣
↪(mit den Seiten des polygon2)
```

```
        #definition der Eckpunkte eines Klotzes a[0]/b[0]/... sind die␣
↪x-Koordinaten der jeweiligen Punkte, a[1]/b[1]/... die y-Koordinaten
        #0 und vier sind die z-Koordinaten. Also ist ein Klotz hier 4LE hoch
        vertices = np.array([\
            [a[0], a[1] , 0], #[-1, -1, -1], #0
            [b[0], b[1] , 0], #[+1, -1, -1], #1
            [d[0], d[1] , 0], #[+1, +1, -1], #2
            [c[0], c[1] , 0], #[-1, +1, -1], #3
            [a[0], a[1] , 4], #[-1, -1, +1], #4
            [b[0], b[1] , 4], #[+1, -1, +1], #5
            [d[0], d[1] , 4], #[+1, +1, +1], #6
            [c[0], c[1] , 4]]) #[-1, +1, +1]]) #7
        #die Flächen des Klotzes werden durch Dreiecke definiert
        faces = np.array([\
            [0,3,1],
            [1,3,2],
            [0,4,7],
            [0,7,3],
            [4,5,6],
            [4,6,7],
            [5,1,2],
            [5,2,6],
            [2,3,6],
            [3,7,6],
            [0,1,5],
            [0,5,4]])

        cube = mesh.Mesh(np.zeros(faces.shape[0], dtype=mesh.Mesh.dtype)) #ein␣
↪Würfel mit den Flächen aus dem Array faces wird erzeugt
        for i, f in enumerate(faces): #schleife läuft über alle einträge im␣
↪faces array
            for j in range(3): #schleife läuft über die drei Eckpunkte eines␣
↪Dreiecks im faces Array
                cube.vectors[i][j] = vertices[f[j],:] #jetzt werden die␣
↪Eckepunkte des cubes mit den Koordinaten aus dem vertices array gefüllt,␣
↪also beschreibt dieser cube jetzt den aktuellen Klotz

        #der cube wird als stl datei gespeichert (dateityp der für den druck␣
↪verwendet wird). Hier ist der Befehl auskommentiert, da sonst beim Ausführen␣
↪der Funktion direkt ohen Nachfrage die stl Dateien der Klötze abgespeichert␣
↪werden
        #cube.save('cube'+str(n)+'.stl')

    #die Funktion der Kettenlinie wird gezeichnet (gleiches Verfahren wie in␣
↪der Funktion zeichnen)
    x_values = np.linspace(-35, 35, 300)
```

```
    y_values = kettenlinie(x_values)

    plt.axhline(0, color='black',linewidth=1)
    plt.axvline(0, color='black',linewidth=1)


    plt.plot(x_values, y_values)
    plt.show()
```

[82]:
```
kloetzeAbspeichern(sekantenLaengenWerte, laengeKloetzeAussenWerteListe,␣
↪zusetzlicheLaengenstueckeWerte, x_werte, y_werte,␣
↪sekantenSteigungswinkelWerteListe)
```



Die Graphik ist verzerrt, führt man das Programm lokal aus, kann man die Fenstergröße anpassen, sodass die Maßstäbe passend sind.

Diese Programmdokumentation kann unter https://andreas-di.github.io/jupyter/lab/index.html aufgerufen werden.